Uncertainty In Service Provisioning Relationships

Thesis by

Chris Smith

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

Newcastle University

Newcastle upon Tyne, UK

2010

"*If there is any society among robbers and murderers, they must at least, according to the trite observation, abstain from robbing and murdering one another.*"

Adam Smith from "The Theory of Moral Sentiments" (1759)

To Mum, Dad, and Nick.

Acknowledgements

I would like to express my gratitude to Professor Aad van Moorsel for his supervision throughout my PhD. The guidance and support that Aad has given are the foundation of this thesis. Additionally, I would like to thank those people I have been fortunate to work with throughout my PhD: Professor Isi Mitrani, Dr Nigel Thomas and Dr Joris Slegers on the EPSRC-funded "Dynamic Operating Policies For Commercial Hosting Environments" project; and Professor Feng Li and Dr Darek Pieńkowski on the EPSRC-funded "Economics-Inspired Mechanisms For Instant Trust In The Service Provision Industry" project. I would also like to express my appreciation for the love and support of my Mum, Dad, my brother Nick, and my girlfriend Kate, who have all been there with me throughout my PhD. Finally, I would like to thank my friends, particularly Simon Woodman, for making the last few years in Newcastle so enjoyable.

Abstract

Web services encapsulate some well-defined functionality such as data storage, computation or a business process. An organization can delegate responsibility for the provision of a Web service to another organization with the formation of a Service Provisioning Relationship (SPR). Such relationships enable organizations to specialize in the provision of those Web services for which they have a comparative advantage, and to outsource responsibility for any other functionality on which they are dependent to the Web services provided by other organizations. This specialization can lead to significant increases in organizational efficiency. The scale of such increases is determined by the extent to which the presence of uncertainty in the Quality of Service (QoS) can be addressed by the organizations in the SPR. The requester of a Web service can be uncertain of the willingness and ability of the provider to provision a Web service with a certain QoS. The provider of a Web service can be uncertain of his ability to provision a Web service to each requester with a certain QoS. These uncertainties can endanger the economic viability of such relationships, and mitigate any increases in organizational efficiency. This thesis provides a number of key contributions to address these uncertainties in order to retain the economic viability of these relationships for both organizations. The key contributions are: an institutional framework for trust in SPRs, a structured language for the representation of a Service Level Agreement (SLA) as a contingent contract, and a theoretical and practical methodology for the creation of SLAs which are optimal with regard to the objectives of an organization.

Contents

List of Figures

Chapter 1

Introduction

Web services encapsulate some well-defined functionality such as data storage, computation or a business process. An organization can delegate responsibility for the provision of a Web service to another organization with the formation of a Service Provisioning Relationship (SPR). Such relationships enable organizations to specialize in the provision of those Web services for which they have a comparative advantage, and to outsource responsibility for any other functionality on which they are dependent to the Web services provided by other organizations. This specialization can lead to significant increases in organizational efficiency. The scale of such increases is determined by the extent to which the presence of uncertainty in the Quality of Service (QoS) can be addressed by the organizations in the SPR:

- **Uncertainty in QoS experienced by the requester of a Web service**
  The requester of a Web service can be uncertain of the willingness and ability of a provider to provision a Web service with a certain QoS. This uncertainty arises from a lack of complete control over, and complete information pertaining to, the Web service. The provider may lack the willingness and ability to provision the Web service to the requester with a certain QoS, or the requester may be unaware of this willingness and ability. Such uncertainty leaves the requester vulnerable to the provision of a Web service with a QoS which does not meet certain requirements, and leads to negative effects on organizational objectives of the requester.

- **Uncertainty in QoS experienced by the provider of a Web service**
  The provider of a Web service can be uncertain of his ability to provision a Web service to requesters with a certain QoS. This uncertainty arises from a lack of complete control over, and complete information pertaining to, the computing and communications infrastructure on which the Web service is provisioned. The infrastructure can be prone to the failure of resources, and fluctuations in the usage patterns of these resources. This introduces variability in the resource capacity and directly affects the QoS. Such uncertainty leaves the provider vulnerable to the provision of a Web service with a QoS which does not meet the requirements

of a requester, and leads to negative effects on organizational objectives of the provider and the requester.

These uncertainties can endanger the economic viability of SPRs, and mitigate any increases in organizational efficiency. The organizations may not participate in SPRs in the presence of such uncertainties, due to the negative effects on organizational objectives from usage or provision of a Web service with a certain QoS. Therefore, the economic viability of SPRs is predicated on the ability of organizations to address such uncertainties. This thesis provides work to address these uncertainties for both organizations in an SPR.

The remainder of the chapter is structured as follows. Section 1.1 provides an example scenario for the purposes of explication in subsequent chapters of the thesis. Section 1.2 summarizes the novel contributions of this thesis. Section 1.3 lists peer-reviewed publications which have been written or co-written by the author containing work which is explicitly included in, or closely related to, the thesis.

1.1 Example Scenario

The example scenario utilized in the thesis is based upon a Web service for the retrieval of stock quotes. The provider of the Web service is an organization named XYZ Stockbrokers, and the requester of the Web service is an organization named ABC Investments. ABC Investments is seeking to increase its organizational efficiency by outsourcing the retrieval of stock quotes to XYZ Stockbrokers. This enable ABC Investments to focus on the provisioning of an Investment Service for which it has a comparative advantage. The Investment Service provides investment advice for the buying and selling of stocks utilizing the quote for a certain stock at a certain point in time, obtained from the Stock Quote Service.

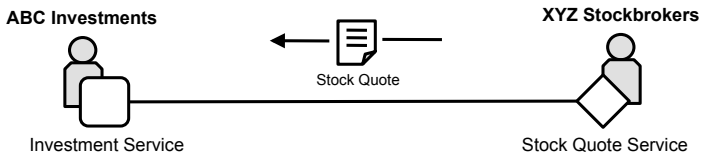

Figure 1.1: An Illustration Of The Example Scenario

In the outsourcing of the Stock Quote Service to XYZ Stockbrokers, ABC Investments requires that the Stock Quote Service is provisioned with a certain QoS: the response time for the retrieval of a stock quote has an upper bound of 200 milliseconds. This QoS enables the Investment Service to provide investment advice based on the most accurate view of the stock market. ABC Investments is uncertain of the willingness and ability of XYZ Stockbrokers to provision to the Stock Quote Service

in accordance with such a QoS due to the inherent lack of control from delegation, and due to a lack of complete information pertaining to, the Stock Quote Service. For example, ABC Investments may have not previously participated in SPRs with XYZ Stockbrokers. XYZ Stockbrokers is uncertain of his ability to provision the Stock Quote Service to ABC Investments in accordance with such a QoS due to a lack of complete control over, and complete information pertaining to, the computing and communications infrastructure on which the Stock Quote Service is provisioned. For example, the computing and communications infrastructure may be subject to significant fluctuations in usage patterns. The economic viability of the SPR is predicated on the ability of ABC Investments and XYZ Stockbrokers to address these uncertainties, such that both expect that participation in the SPR is consistent with organizational objectives.

1.2 Thesis Contributions

This thesis provides a number of key contributions to address the problem of uncertainty in SPRs:

- **A cross-disciplinary background and literature review of uncertainty in SPRs**
  This cross-disciplinary background and literature review summarizes the key concepts and methodologies from a variety of different disciplines including computer science, economics and sociology. Such an approach differs from previous work, which focuses on concepts and methodologies from the perspective of a single discipline. The value of this cross-disciplinary approach is to provide of a theoretical and practical basis for the study of uncertainty in SPRs which jointly considers technological, economic and social perspectives. This enables such perspectives to inform the design and evaluation of software components to support SPRs.

- **An institutional framework for trust in SPRs**
  This framework provides a holistic representation of the environment of an SPR, and the perception of this environment by an organization to derive a notion of trust. Such an approach differs from previous work, which makes implicit or explicit assumptions regarding the environment without explanation of the implications of such assumptions for trust, and often focuses on environments containing a single class of trust mechanism rather than environments as a holistic encapsulation of the context of an SPR. Additionally, in contrast to previous work, institutions are utilized to model the constraints in the environment of an SPR, such that institutions are perceived by an organization to analyze, evaluate, and decide upon participation in an SPR. The value of this approach is to provide a basis for the evaluation of an environment of an SPRs in terms of the absence or presence of trustworthiness and trust, and the mismatch between trust and trustworthiness. This enables environments to be appropriately designed or manipulated by organizations to increase both trustworthiness and trust.

- **A structured language for the representation of SLAs**
  This structured language enables the representation of SLAs as contingent contracts, whose contingencies are defined over the QoS experienced by the requester. Such an approach differs from previous work, which does not provide a formal basis for the representation of SLAs as contingent contracts, where contingencies are defined in a verifiable and enforceable manner utilizing HTTP status codes. The value of this approach is to enable the utilization of insights from economic literature on the design of contingent contracts to the creation of SLAs, such that the uncertainty of an organization in the QoS can be reduced.

- **A theoretical and practical methodology for the creation of optimal SLAs**
  This methodology facilitates the creation of SLAs as contingent contracts, whose contingencies are optimal with the objectives of an organization, in the presence of uncertainty in QoS. Such an approach differs from previous work, which does not provide a generic process for the creation of optimal SLAs for any QoS metric of a Web service, given organizational objectives, and does not describe the practical methodology required to realize such an optimization process in a software component. The value of such an approach is to provide a formal process for the creation of optimal SLAs, such that organizations no longer utilize processes based on ad-hoc, sub-optimal reasoning which can be inconsistent with organizational objectives.

1.3 Publication History

This thesis includes work which has been previously included in, or is closely related to, peer-reviewed publications which have been written or co-written by the author. These publications are referenced appropriately in subsequent chapters of the thesis, and are listed as follows:

- Feng Li, Darek Pieńkowski, Aad van Moorsel and Chris Smith. A Holistic Framework For Trust In Online Transactions. *International Journal Of Management Reviews* (Submitted). Wiley, 2010.

- Ravi Teja Dodda, Chris Smith, and Aad van Moorsel. An Architecture For Cross-Cloud System Management. In Sanjay Ranka, Srinivas Aluru, Rajkumar Buyya, Yeh-Ching Chung, Sandeep Gupta, Ananth Grama, and Rajeev Kumar, editors, *International Conference On Contemporary Computing*, Communications in Computer and Information Science. Springer, August 2009.

- Chris Smith, Aad van Moorsel, Feng Li and Darek Pienkowski. An Institutional Approach To Trust In Electronic Transactions. *International Conference on the Technical and Socio-Economic Aspects of Trusted Computing (Trust'09)*, Oxford, UK, April 2009.

- Chris Smith and Aad van Moorsel. Mitigating Provider Uncertainty In Service Provision Contracts. In D. Neumann, M. Baker, J. Altmann, and O.F. Rana, editors, *Economic Models and Algorithms for Distributed Systems*, Autonomic Computing. Birkhäuser, February 2009.

- Chris Smith and Aad van Moorsel. Representational State Transfer Based System Management. In H.G. Hegering, H.Reiser, M.Schiffers, and Th.Nebe, editors, *Proceedings of the 14th Annual Workshop of HP Software University Association (HP-SUA)*, Garching/Munich, Germany, July 2007. Managebright.

- Emerson Ribeiro de Mello, Savas Parastatidis, Philipp Reinecke, Chris Smith, Aad van Moorsel, and Jim Webber. Secure And Provable Service Support For Human-Intensive Real-Estate Processes. *In Proceedings of the IEEE International Conference on Services Computing (SCC'06)*, pages 495-502, Chicago, USA, September 2006. IEEE Computer Society.

The remainder of this thesis is structured as follows. Chapter 2 presents a background and literature review to provide the context for this thesis. Chapter 3 presents an institutional framework for trust in SPRs. Chapter 4 presents a software architecture for SPRs. Chapter 5 presents an empirical study of uncertainty in SPRs. Finally, the thesis is briefly summarized and concluded in Chapter 6.

Chapter 2

Background And Literature Review

This chapter presents a background and literature review for the thesis, and provides the context for the problem of uncertainty in SPRs. The problem of uncertainty in QoS is explored for the requester of a Web service, with consideration of trust; and for the provider of a Web service, with consideration of management. The fundamental concepts utilized in the thesis are introduced, and the relationships between these concepts is explored. The relevant literature pertaining to these concepts is then reviewed, drawing from a number of different disciplines including computer science, economics and sociology. Such an approach differs from previous work, which focuses on concepts and methodologies from the perspective of a single discipline. The value of this cross-disciplinary approach is to provide of a theoretical and practical basis for the study of uncertainty in SPRs which jointly considers technological, economic and social perspectives. This enables such perspectives to inform the design and evaluation of software components to support SPRs.

The remainder of the chapter is structured as follows. Section 2.1 introduces organizations, Web services and SPRs. Section 2.2 discusses QoS with regard to Web services. Section 2.3 introduces uncertainty in QoS. Section 2.4 describes the utilization of SLAs to explicitly define QoS. Section 2.5 explores uncertainty for the requester in an SPR, describing trust in QoS and reviewing the different trust mechanisms. Section 2.6 explores uncertainty for the provider in an SPR, describing management of QoS and reviewing different management mechanisms.

2.1 Organizations And Web Services

An organization is a collection of one or more "real-world people" [1] with a set of capabilities. These capabilities are coordinated within an organization to pursue certain objectives, such as maximum revenue. The capabilities of an organization can be encapsulated within functional units which are "desirable from some person or organization's point of view" [1], and can include well-defined functionality such as data storage, computation or a business process. Such functional units can

be exposed to these organizations as Web services which enable "machine-to-machine interaction over a network" [1]. The provider is the organization whose capabilities are encapsulated within the Web service, and the requester is the organization which utilizes these capabilities through the Web service.

A Service Provisioning Relationship (SPR) is a dependent relationship formed between a requester and provider for the provision and subsequent usage of Web service. Such relationships can be referred to with alternative terminology including: delegation relationships [1], outsourcing relationships [2], and agency relationships [3, 4]. The terminology utilized is dependent on the discipline in which the relationships are studied. Any such relationship refers to the delegation of responsibility for the provision of certain functionality from a requester to a provider for some period of service usage. This enables the requester to specialize in the provision of capabilities and, therefore, Web services, for which they have a comparative advantage. Any other functionality on which they are dependent can be outsourced to the Web services provided by other organizations. Such specialization and division of labor [5] is a fundamental tenet of economic efficiency.

There are a number of innovative organizational paradigms [6, 7] which exploit SPRs, such as "Software as a Service" (SaaS) and "Infrastructure as a Service" (IaaS). These paradigms "separate the possession and ownership of software and hardware from its use" [8], enabling hardware and software providers to specialize in their specific capabilities, and other organizations to exploit these capabilities. Current commercial examples of such services include Amazon Elastic Compute Cloud (EC2) [9], Microsoft Azure [10] and Google App Engine [11].

2.1.1 Agents

An agent is a "piece of software or hardware that sends, receives and processes messages" [1]. Such agents are utilized to represent the organizations which interact through a Web service. A service interface defines the interactions supported by a Web service, stipulating the types and patterns of messages, and the semantics of the functionality provided by the Web service. The requester utilizes an agent to interact with the provider through the service interface and utilize the functionality of a certain Web service. The provider utilizes an agent to interact with the requester through the service interface and provision the functionality of a certain Web service.

A rational agent is an agent whose interactions reflect the unilateral, self-interested pursuit of organizational objectives. Such a model of behavior is commonly utilized in economic literature, and provides a sound, well-studied basis for modeling the behavior of organizations, and their respective agents. A rational agent encapsulates a system of practical reasoning which involves "weighing conflicting considerations for and against competing options, where the relevant considerations are provided by what the agent desires/values/cares about and what the agent believes" [12]. The BDI (Beliefs Desires Intentions)-Model [12, 13] is often utilized to model the process of practical reasoning

of an agent, where the beliefs of the agent represent its current informational state, the desires of the agent represent its objectives, and the intentions of the agent represent the commitment to a certain policy in order to pursue objectives given beliefs and attitudes. The attitudes define the manner in which interests are pursued given beliefs, and therefore the intentions which emerge from the desires of the agent.

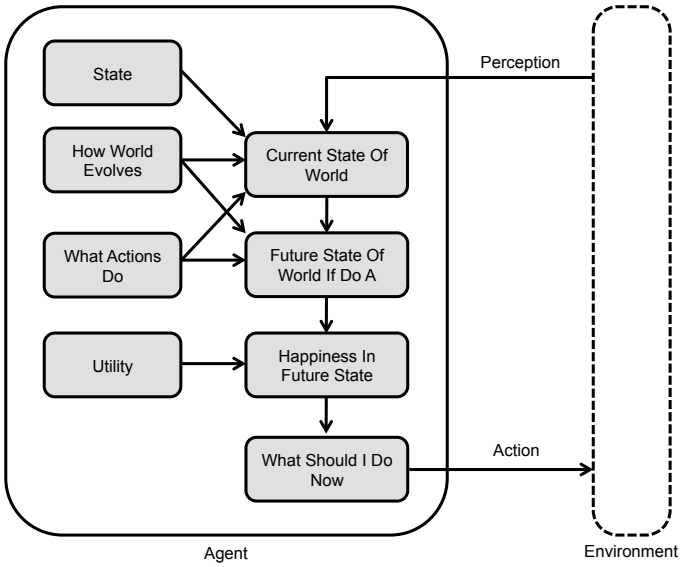

Figure 2.1: An Illustration Of A Utility-Based Agent Architecture

Figure 2.1 illustrates an abstract agent architecture which realizes a certain process of practical reasoning for an agent as a software system. There are many different classes of agent architecture including [14]: reflex agents, model-based agents and goal-based agents. These architectures vary in the manner with which the reasoning process is represented by software components, and the functionality of these software components. The agent architecture shown is a utility-based agent (adapted from [14]), which formulates policies based upon a set of beliefs about the environment and a set of objectives, encoded as a utility function over the current and future states of the environment. The policy stipulates those interactions which are deemed to yield the greatest utility, in expectation, where expected utility is the average utility across all feasible future states of the environment, weighted by expectation. For example, the requester of a Web service may compute a policy which stipulates the provider with whom to participate in a SPR, given beliefs about the willingness and ability of a set of providers for a Web service.

A "bounded rational" [15, 16] agent has a process of practical reasoning which is bounded by finite computational, communication and storage resources. This can lead to the presence of incomplete and imperfect information within beliefs, and sub-optimality within the reasoning process of the agent. Such bounded rationality is particularly common in multi-agent environments, where the

environment with which an agent interacts consists of other agents. The ability to make optimal decisions in such an environment is predicated on the ability to characterize the behavior of the other agent(s) in the environment, who may also be rational. In order to characterize such behavior, the agent must be able to characterize the reasoning process of the other agents(s), which includes beliefs about the agent which is attempting to characterize their reasoning process. Such beliefs can become highly inter-related, and lead to an infinite regression of reasoning over beliefs, with agents holding increasingly higher-order beliefs pertaining to other organizations, *ad infinitum*. Such intractability in reasoning motivates the utilization of bounded forms of rationality which place restrictions on the order of beliefs and accept the presence of imperfect or incomplete beliefs, and the ubiquity of theoretically sub-optimal policies amongst agents.

2.1.2 World Wide Web

The World Wide Web (WWW) is an "information space" [17] in which agents interact with items of interest, or resources, which are uniquely identified by a Uniform Resource Identifier (URI) [18]. These resources reside under the control and ownership of a certain organization, and can be Web services which encapsulate certain functionality. The interaction of agents for such resources is performed over the communication network provided by the Internet. A communication network consists of a set of agents connected by communication channels in some topology, where a communication channel is a bilateral connection for communication and interaction between two agents. Such interaction utilizes standardized communication protocols to define a set of rules for the syntax, semantics and ordering of messages. Hypertext Transfer Protocol (HTTP) [19] is commonly utilized as the protocol for interaction amongst agent for resources on the World Wide Web.

The Internet is an open, distributed system which consists of a vast number of interconnected physical communication channels. These communication channels reside under the control/ownership of organizations in different social, political, administrative and geographical domains. The interaction of agents for a resource on the World Wide Web utilizes a logical communication channel on the Internet which is composed of a number of physical communication channels. These physical communication channels together provide a path for communication between the agents. Accordingly, interaction over the Internet is subject to the control of all organizations whose physical communication channels are utilized by a logical communication channel between agents. These organizations can control the capacity of the physical communication channel assigned to any logical communication channel, and therefore determine the time taken to perform an interaction or whether or not the interaction is performed. The "random and non-negligible communication delays and partial failures" [20] which may arise on the Internet provide "fundamental limits to the reliability of communication" [1] between agents on such a network.

Figure 2.2 illustrates the structure of a communication channel on the World Wide Web. ABC

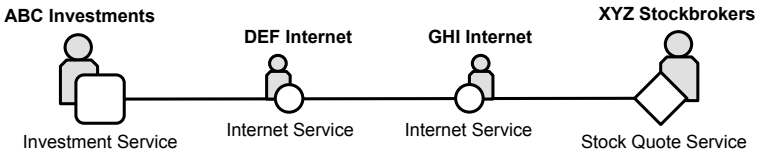

Figure 2.2: An Illustration Of The World Wide Web

Investments interacts with the resource or Web service of XYZ Stockbrokers over a logical communication channel which utilizes physical communication channels under the control of two different organizations: DEF Internet and GHI Internet. These organizations provide the Internet Services with which ABC Investments and XYZ Stockbrokers communicate. These Internet Services can control the capacity assigned to the logical communication channel in accordance with their own objectives and determine the time taken to interact with the Stock Quote service or whether or not ABC Investments can interact with the Stock Quote service.

2.2 Quality Of Service

Quality of Service (QoS) refers to the non-functional properties of interaction with a Web service. These properties determine the effectiveness with which the functionality of the Web service is provisioned, and define the extent to which a Web service is able to "satisfy stated or implied needs" [21] of a requester. Such properties can "determine a software product's success or failure" [22]. QoS can broadly grouped into three categories [23, 24, 25]: performance, availability and security. These categories contain a set of different non-functional properties, or metrics. A metric defines a non-functional property of a Web service in a quantifiable manner in order such that unambiguous semantics can be associated with that property. A service level represents a concrete instantiation of metrics with their respective values. This service level can contain actual values obtained from the measurement of metrics, reflecting past service usage, or can contain expected values reflecting future service usage.

The measurement of QoS metrics must consider [26]: which metrics are to be measured, the means by which they metrics can be measured, which organization performs the measurement, and the point in the communication network at which the measurements are performed. The requester and provider have different perspectives on the Web service and are concerned with different types of metrics [27]:

System metrics System metrics are those metrics which define non-functional properties of the service from the perspective of the provider. The value of system metrics reflects the state of the computing and communications infrastructure on which the service is provisioned. Examples of such metrics include: queue size for interactions and failure rate of interactions. These metrics are

conventionally associated with the management of QoS. The provider can evaluate capabilities of Web services based upon different actual or expected values of these system metrics, and perform management on the computing and communications infrastructure based on such evaluation.

Experience metrics Experience metrics are those metrics which define non-functional properties of a Web service from the perspective of the requester. The value of experience metrics reflects the subjective experience of the requester from service, and can be referred to as Quality of Experience (QoE) metrics. Examples of such metrics include: response time of interaction and availability of the service for interaction. These metrics are conventionally associated with trust in QoS. The requesters can "evaluate requirements" [28] of Web services based upon the different actual or expected values of these experience metrics, and differentiate amongst the Web services based on such evaluation. These metrics are influenced by both the Web service and the communications channel on which the requester interacts with the Web service.

Business metrics Business metrics are those metrics which define the objectives of the requester and provider in terms of the non-functional properties of the Web service, and can be referred to as Quality of Business (QoBiz) metrics. The business metrics of the requester are defined in accordance with experience metrics from which they derive value, and the business metrics of the provider are defined in accordance with system metrics from which they incur costs. Examples of such metrics include: revenue and costs. An organization can evaluate the extent to which the usage or provision of a Web service is consistent with their objectives based upon different actual or expected values of these business metrics.

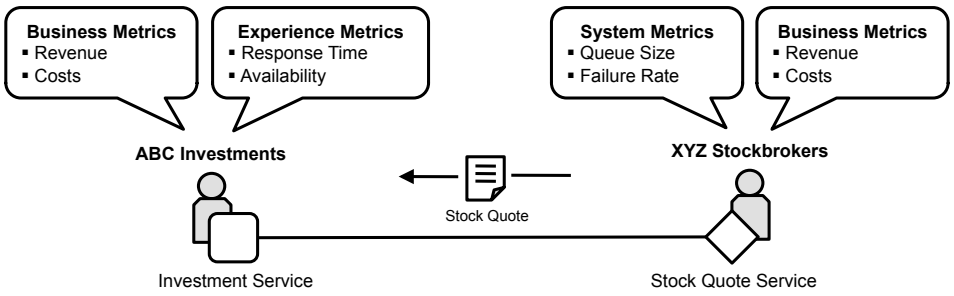

Figure 2.3: An Illustration Of QoS Metrics

Figure 2.3 illustrates the different types of QoS metrics with which the organizations in the SPR are concerned, and provides some examples of these different metrics. The business metrics of ABC Investments are determined by experience metrics, or the QoS at the point of consuming the Web service. The business metrics of XYZ Stockbrokers are determined by system metrics, or the QoS at the point of provisioning the Web service.

2.3 Uncertainty

Uncertainty is a pervasive concept across a vast number of phenomena, and arises from the presence of change [29]. The ability to make decisions [30] or judgements [31] with regard to the current or future state of phenomena is directly affected by the presence of uncertainty. There exist many definitions of uncertainty which are specific to the phenomena in which the uncertainty arises. In relation to the QoS of a Web service, uncertainty can be defined as the inability to assign a value to a metric with certainty. This uncertainty can be partitioned into two distinct classes which differentiate the source of the uncertainty [32, 33]:

Aleatory uncertainty Aleatory uncertainty is randomness whose source is an absence of control, or inherent randomness, in the value of a metric. This can be alternatively termed as objective uncertainty, non-variable uncertainty or "theoretical ignorance" [14]. For example, the value of a metric can be subject to aleatory uncertainty due to absence of control over the usage patterns of a Web service, where the Web service is provisioned on a finite capacity computing and communications infrastructure whose resource capacity is partitioned across concurrent requesters.

Epistemic Uncertainty Epistemic uncertainty is randomness whose source is inaccuracy in the model of a value from the perspective of an organization. This can be alternatively termed as subjective uncertainty, variable uncertainty, or "practical ignorance" [14]. For example, the value can be subject to epistemic uncertainty due to the inability of an organization to enumerate and objectively represent all influencing factors on the value.

Uncertainty is commonly modeled and reasoned over utilizing Probability Theory [34]. This mathematical framework enables the construction of a probability model, which consists of a set of independent and/or causally linked random variables. These random variables model the individual, influencing factors on the value of a metric which are subject to uncertainty, and combine to model the uncertainty in the value of a metric expressed by an organization at a given point in time. For example, a probability model for the availability of a Web service may comprise of the three random variables X, Y and Z, where Z is the availability of the service, and is causally linked to from X and Y variables, where X represents the failure of software components in the computing and communications infrastructure and Y represents the failure of hardware components. This model can be utilized to evaluate assertions with regard to the value of a metric to derive a probability or "degree of belief" [14] in that assertion, defined as a real value in the interval $[0, 1]$.

There are two different classes of random variable: discrete and continuous. A discrete random variable can represent a metric with a countable number of values, and has an associated probability mass function which defines the probability of the variable assuming each of these values. For

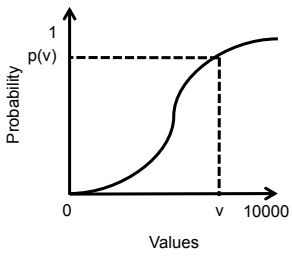

Figure 2.4: An Illustration Of A Continuous Probability Distribution

example, a discrete random variable, X, representing the availability of a Web service may assume a value in: $\{true, false\}$, and have an associated probability distribution defining the probability of the service being available or unavailable. A continuous random variable can represent a metric with an uncountable number of values, and has an associated probability distribution which defines the probability of the variable assuming a value in a given interval of values. Figure 2.4 illustrates a probability distribution for a continuous random variable representing the time taken to perform an interaction with a Web service, which may assume a value in: $[0, 10000]$. The probability that the value of a metric is below a certain value, v, can be derived through integration over the area under the curve up to v. Both classes of random variable have a mean or expected value, $E(\cdot)$, and a variance or spread of values, $Var(\cdot)$, which are dependent on the probability density functions (PDF) and cumulative distribution functions (CDF) associated with the random variable.

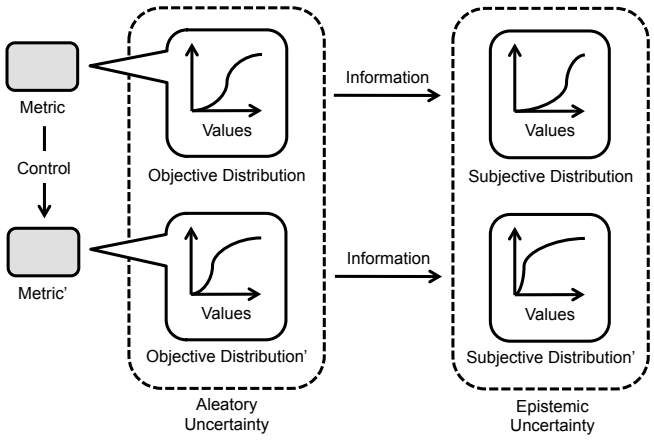

Figure 2.5: An Illustration Of Probability Theory To Represent Uncertainty

Figure 2.5 illustrates the different classes of uncertainty utilizing Probability Theory. The aleatory uncertainty in the value of a metric is represented by an objective probability distribution over domain of values for the metric. This represents a perfect model of the metric. Aleatory uncertainty can be reduced with augmented control of influencing factors on the value of the metric, manipulating these factors and the objective probability distribution over these values. For example, control could be imposed to reduce the aleatory uncertainty in the value of a metric. The epistemic

uncertainty is represented by a subjective probability distribution. This represents a model of the value of a metric which is subject to inaccuracies. Epistemic uncertainty can be reduced with more effective modeling methods, which more realistically include the influencing factors on the value of the metric. This derives a subjective probability distribution which more accurately represents the underlying objective probability distribution. Typically, these two classes of uncertainty coexist within a phenomenon and the extent to which uncertainty can be reduced is predicated on the extent to which the phenomenon can be controlled or the extent to which information on the phenomenon is available. For example, in the absence of an ability to control, epistemic uncertainty is the only uncertainty which can be reduced.

2.3.1 Costs Of Uncertainty

In the presence of uncertainty, "all methods of reducing or redistributing uncertainty involve costs" [29]. These costs are attributable to the opportunity cost of resources which are consumed in the reduction or redistribution of uncertainty, such as time and money. The extent to which costs will be tolerated to reduce uncertainty is "based upon how far uncertainty is considered undesirable" [29], and the benefits which can be yielded from the reduction in uncertainty. Higher costs will be tolerated to reduce aleatory and epistemic uncertainty in a Web service for which the QoS is considered mission-critical. The presence of uncertainty is highly undesirable in such Web services and high benefits can be yielded from the reduction of uncertainty.

The costs of uncertainty in an SPR can be referred to as "transaction costs" [35, 36]. These transaction costs represent the costs incurred to "sustain desirable activities or to stop undesirable activities" [37] in a dependent relationship which is subject to the presence of uncertainty. Transaction costs can "impede and in particular cases can completely block the formation of markets" [3], where the uncertainty is prohibitively costly for organizations to overcome. Indeed, the complete failure of a market represents the existence of prohibitive transaction costs, such that their incurrence within an SPR would negate any efficiency gains from the relationship, and the SPR is no longer economically viable for organizations. These costs can be divided into three different classes [35]:

Search and Information Costs These are the transaction costs associated with the search for requesters and/or providers of a particular Web service with a certain QoS. For example, the costs associated with the gathering and processing of information pertaining to potential providers of a Web service.

Negotiation and Contracting Costs These are the transaction costs associated with the negotiation between the requester and provider for the provision and utilization of a chosen Web service

with a certain QoS, and the contracting which arises from this negotiation. For example, the costs associated with utilization of a third party to verify commitment of the organizations to a contract.

Policing and Enforcement Costs These are the transaction costs associated with the monitoring and enforcement of contracts formed for a chosen Web service with a certain QoS. For example, the costs associated with the utilization of a third party for the monitoring of QoS.

2.4 Service Level Agreements

A Service Level Agreement (SLA) expresses the "functional and non-functional properties of Web services in a declarative manner" [23]. It stipulates the "obligations and expectations" [38] of organizations in an SPR to provide a "shared expectation about the interaction with the Web service" [1]. An SLA contains of a set of Service Level Objectives (SLOs), which represents the assignment of an expected value to a metric. This set of SLOs defines the QoS that can be expected from the Web service. A set of SLOs may define a class of QoS such as Gold, Silver and Bronze [39] or Guaranteed and Best-Effort [40]. These classes may contain expected values for one or more metrics, such as availability and response time. The SLOs contained within an SLA reflect a process of negotiation between the requester and provider, which finds mutuality in the requirements of the requester and the capabilities of the provider. Such a negotiation process for SLAs is described in [41]. The SLA can associate monetary obligations with SLOs such that monetary payments are made between the organizations for the provision of the Web service with a QoS defined by a set of SLOs. Figure 2.6 summarizes the key elements within a SLA in accordance with [2, 23, 38].

An SLA can be defined in a "machine processable or human oriented" [1] manner. The elements in Figure 2.6 have been utilized, under varied terminology, in a number of different ontologies and languages for SLAs. WS-Agreement [39] is the most notable language, whose goal was to "standardize the terminology, concepts, and overall agreement structure" [39] to enable a common understanding amongst organizations. The expressiveness of WS-Agreement over functional and non-functional properties has been extended in work such as [42]. An alternative language is SLAng defined in [43], which provides a language for the expression of SLAs which are specific to different types of Internet service usage. Cremona [44] is a software architecture devised to provides support for the creation and monitoring of such SLAs defined utilizing WS-Agreement such that the requester and provider can automate creation and negotiation of SLAs. WS-Agreement provides standardized terminology and structure for SLAs but does not define an ontology for the unambiguous definition of QoS metrics.

WS-Policy [28] provides a language for the expression of QoS as set of assertions which must hold for the Web service. The language can be utilized with WS-Agreement to define the QoS of a Web

Name	Description
Purpose	The reason for the agreement between the organizations.
Parties	The organizations engaging in a SPR.
Validity Period	The period of the agreement defined with a start time and end time or number of invocations.
Scope	The set of Web services whose properties are covered by the agreement.
Restrictions	The obligations of the requester to receive the specified Web service.
Service Level Objectives	The QoS stipulated as a set of metric and a target values for those metrics.
Penalties	The sanctions applied in the event that the SLOs are not fulfilled.
Optional Services	The Web services which may be required on the occurrence of an exception or undefined event.
Exclusion Terms	The properties of the Web service which are not covered by the agreement.
Administration	The processes by which metrics will be observed and the organizations responsible for observation.

Figure 2.6: A Table Of Elements In An SLA

service as disjunctions and conjunctions of such assertions. This language provides an arbitrarily rich set of expressions of QoS. The language is focused on security properties of a Web service and does not introduce a specific vocabulary of metrics for performance or availability metrics of Web services, over which requirements and capabilities may be expressed by organizations. An ontology is presented in [45] to enable the expression of a number of different QoS metrics in a commonly understood manner. DAML-QoS [46] is an alternative ontology which enables the specification of such QoS metrics in an unambiguous and commonly understood manner amongst organizations.

2.5 Trust

Trust is a concept which has been defined in a variety of literature across a number of different disciplines. An overview of trust and related definitions is provided in [47], including definitions for concepts such as mistrust and distrust which are omitted from consideration in this thesis. The definition we choose to utilize in the thesis is a subjective belief in the trustworthiness of a Web service. It represents a "device for coping with the freedom of others" [48] to interact in any manner, and the uncertainty which arises from such freedom, enabling an SPR to be formed between organizations in the presence of uncertainty. There are notions of "predictability" [48, 49] and "vulnerability" [50] simultaneously encapsulated within trust: a higher level of trust simultaneously represents a higher perceived predictability and lower perceived vulnerability.

Trustworthiness refers to the "competence of an entity to act dependably, securely and reliably" [51] and is closely linked to the concept of dependability, which pertains to "the ability to deliver a service that can be justifiably trusted" [52]. The notions of dependability, security and reliability to which trustworthiness pertain can be simultaneously encapsulated within the concept of QoS. This competence is interpreted through a process of "subjective reasoning" [53] to enable trust to be "quantified by a subjective probability" [54]. The "particular level of the subjective probability" [55] represents a belief that the Web service "do what it says it will (being honest and reliable)" [56, 57] and "provide a specific service" [51] with a certain QoS. This derivation of the subjective probability arises "both before he can monitor such action (or independently of his capacity ever to be able to monitor it) and in a context in which it affects his own action" [55].

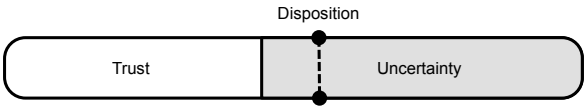

Figure 2.7: An Illustration Of Trust, Uncertainty And Disposition

Figure 2.7 illustrates the distinction between trust and trustworthiness. A Web service may have a certain trustworthiness, in terms of a competence to provide a certain dependability, security or

reliability, but this may differ from the trust which is placed in the Web service by an organization. The subjectivity of the reasoning process by which trust is derived provides opportunities for varied interpretation of the competence, and therefore varied trust amongst different organizations for a Web service with a certain trustworthiness. Disposition [58, 59] refers to the level of tolerable uncertainty for an organization, and determines the trust which is necessary to establish a dependency on another organization for a Web service. This disposition is completely subjective and reflects the attitudes of the organization and the manner in which policies are formulated to pursue objectives given these attitudes and the beliefs of the organization. In the presence of a level of trust which exceeds "some personal threshold (the level of acceptable trust)" [54], or some disposition, the organization chooses to establish a dependency on another organization for a Web service.

Trust represents the belief in the trustworthiness of a Web service "within a specified context" [51], such that trust is "context-based" [51] and "situation-specific" [60]. The trust of an organization in a certain context or environment "doesn't directly transfer to another environment" [60]. In dynamic environments, whose properties "change rapidly" [61], organizations must "monitor trust relationships" [51] with other organizations in order to "determine whether the criteria on which they are based still apply" [51]. These trust relationships must be "adjusted to reflect recent interactions" [62] and the outcomes of these interactions. This requires a "high level abstract way of specifying and managing trust" [51] to enable the "monitoring and re-evaluating existing trust relationships" [51] between organizations. In dynamic and large scale environments, organizations may have "no prior interactions" [63] such that organizations have "insufficient information about the service provider and the services offered" [64]. Such an absence of information may lead to an inability of organizations to "distinguish between high and low quality service provider organizations" [64] and to manage trust relationships in a meaningful manner. The trust which arises in the absence of any experience or first-hand knowledge between organizations is often termed "initial trust" [58] or "swift trust" [65], and involves the utilization of rapid cognitive cues [65], generalizations and heuristics.

There are a variety of different mechanisms, or trust antecedents, which are utilized organizations to establish trust with other organizations. These mechanisms have been surveyed in the context of SPRs in a variety of different literature including [51, 57, 64]. Such mechanisms have been broadly classified into "individual level mechanisms" [57] and "system level mechanisms" [57] based upon whether or not information which is specific to an organization is utilized to establish trust in that organization. The individual level mechanisms focus on the "intrinsic properties" [66] of organizations. These properties pertains to the "honesty or reciprocative nature" [57] of an organization, such that trust is managed on a per-organization basis. This can lead to a prohibitive cost in the management of such trust in dynamic and large-scale environments where there exist a high number of organization with whom an SPR may be formed. The system level mechanisms focus on the "extrinsic properties" [66] of the environment in which an SPR is formed. These properties pertain to

the ability of an "overseer with his whip to do his bit" [50], forcing an organization to be trustworthy by "rules of encounter" [57] which define "guarantees, safety nets and other performance structures" [67, 68]. This enables trust to be managed on a cross-organization basis such that trust can be formed between organizations with limited or no information. The trust which formed through such mechanisms has been referred to as "control trust" [69] or "institutional trust" [49, 58, 59, 67, 70]. This reliance on external control has been referred to as the "antithesis of trust" [48] between organizations, with such control structures working "against formation of trust within group or result in the atrophy of already existing trust" [71].

2.5.1 Trusted Third Parties

A Trusted Third Party (TTP) is an organization which "correctly and honestly" [72] provides certain functionality. The TTP fulfills a "particular purpose" [73] in a trustworthy manner, and is trusted in this purpose by other organizations. A TTP can have many roles in a dependent relationship between organizations such as an SPR. These different roles are discussed further in [74, 75, 76, 77, 78], and each seek to reduce the uncertainty of organizations in a certain aspect of the relationship. The explicit modeling of these roles within an SPR is explored in [63]. Broadly speaking, TTPs can be an intermediary for every interaction between organizations in the relationship, or can be involved on the occurrence of an interaction with certain properties, such as exceptional circumstances. The economic viability of such TTPs arises from the different classes of transaction cost (see Section 2.3.1) which emerge in an SPR in order to reduce uncertainty in QoS, and the willingness of the organizations to incur such costs. These costs provide business opportunities for the provision of TTPs, as discussed in [79], and demonstrates the existence of "markets for trust production" [49].

A TTP can be realized with "trusted computing" [73, 80] modules. These modules represent tamper-resistant software and hardware, whose functionality is certified to be trustworthy by another TTP. Such hardware and software can be provisioned within an organization which itself is not trusted by other organizations. For example, such modules are utilized in [81] to perform trusted monitoring functionality within a non-trusted organization. A TTP can be implemented in a distributed manner utilizing Secure Function Evaluation (SFE) [82, 83, 84, 85, 86]. SFE utilizes cryptography to ensure the correct evaluation of a certain function by a set of organizations "containing an honest majority" [86] and malicious minority. This enables the functionality required from a TTP to be provided by a set of mutually distrusting organizations, and enables the detection of those organizations in the set which deviate from this functionality. The adherence to the functionality is based on the fact that an organization cannot gain from such deviation and is "prohibited from acting against the granted interests of other principals it serves." [87]. This is closely linked to the notion of "faithfulness" [88, 89, 90], in which organizations are assumed to be rational, rather than purely malicious or honest. Therefore, instead of assuming that there exists an honest majority of

honest organizations and a minority of malicious organizations, it is assumed that all organizations are rational and will deviate given perceived benefit from such deviation.

2.5.2 Authentication Mechanisms

Authentication mechanisms enable the veracity of claims regarding organizations or Web services to be proved. This verification yields one or more tokens which encapsulate a set of claims whose veracity has been proved. These tokens define a binary notion of truth for the claims, and can be distributed to organizations in the event that a claim must be verified. A Certification Authority (CA) is the organization which issues tokens to other organizations. For example, Verisign [91] is a CA which issues tokens to verify the identity of a real-world organization. Such mechanisms are often bootstrapped on real-world verification of claims, utilizing real-world infrastructure to verify claims off-line, and simply enable tokens representing such verification to be distributed online. There are two classes of authentication mechanism [92, 93]:

Centralized Authentication Centralized authentication utilize a single source of information to verify claims pertaining to an organization or Web service. X.509 [93, 94] is an example of centralized authentication, with hierarchical trust relationships where digital certificates are issued by a single certification authority.

Decentralized Authentication Decentralized authentication utilize multiple sources of information to verify claims pertaining to an organization or Web service. PGP [93] is an example of decentralized authentication, with decentralized trust relationships constructed through transitive "trust paths" [95] between organizations.

Authentication is a "hard security mechanism" [61, 64] based upon a "static notion of credentials and permissions" [60]. The "waterproof protection" [61] provided by authentication can pertain only to those organizations whose claims focus on static credentials, such as identity. Those claims which are subject to dynamism, such as claims pertaining to dynamic properties of an organization or Web service cannot be easily verified to provide such protection. Additionally, such protection is predicated on the presence of organizations within a certain domain, where there exists one or more CAs with the ability and authority to verify claims over these organizations. In order to "support internet-scale transactions" [96], where organizations can reside is varied domains, there must exist a "mapping of organizations across multiple domains" [1]. Such a mapping would enable those tokens which verify the claims of an organization in one domain to be valid, for the purposes of verification, to an organization in another domain.

2.5.3 Learning Mechanisms

Learning mechanisms enable trust in an organization to be formed from the repeated interaction with that organization. An organization interacts with another organization, gathers information pertaining to the trustworthiness of the organization, and processes the information to derive, and subsequently refine, a belief in the trustworthiness of that organization. This feedback loop [60, 97] can incorporate techniques such as reinforcement learning or neural networks [14] in order to provide a formal basis for the process of learning. Such mechanisms provides an "opaque" [98] belief in the trustworthiness an organization which often omits contextual factors which influence the interaction of the organization. Such trust is often referred to as "familiarity" [99] or "knowledge-based" [100] trust.

This process of learning can be costly for organizations, with organizations having to "accept the risk of being abused for learning purposes" [54]. The process is also prone to the "asymmetry" [54] principle, where belief in an organization can be gradually increased through repeated interaction but instantly destroyed with a single interaction. The requirement for a process of learning can also prevent a relationship from occurring between "completely trustworthy" [54] organizations, but who lack sufficient information pertaining to each other for trust to exist. Such an absence is likely to occur in environments where there exists a high number of organizations whose relationships may be short and infrequent.

2.5.4 Socio-Cognitive Mechanisms

Socio-cognitive mechanisms [98, 101] enable trust in an organization to be formed from the cognitive and social properties of an organization. These mechanisms provide a "theory of the mind" [98] for an organization, incorporating the potentially complex structure of beliefs, attitudes and objectives, and reflecting aspects such as personality, shared values, morality and goodwill. Such a behavioral model enables the "competence, willingness, persistence and motivation" [57, 98] of an organization to be evaluated within a certain context, such that the behavior of an organization within a specific context can be predicted. This enables the cognitive process of the organization to be separated from the context in which the organization interacts, such that the behavior of a certain organization can vary with context. Trust formed in this manner is often termed "cognitive trust" [58, 59, 102].

These mechanisms require in-depth knowledge of the beliefs, attitudes and objectives of an organization in order to enable their behavior to be characterized by an appropriate behavioral model. Such knowledge is likely to be costly to obtain and infeasible to hold for high numbers of organizations. A more feasible approach is to utilize a general socio-cognitive model for organizations, such that any organization can be assumed to conform to that socio-cognitive model, and behave in a certain manner within a certain context. For example, an organizations may be assumed to

be malicious, honest or rational. This provides a more tractable process of trust management for dynamic and large-scale environments, whilst providing a sound and generic basis for predicting behavior in a certain context. Such assumptions of general models are utilized as the building blocks of many other types of mechanisms such as reputation mechanisms and contract mechanisms. For example, a contact mechanism only provides control of organization under the assumption that the organization is rational rather than malicious.

2.5.5 Reputation Mechanisms

Reputation mechanisms enable trust in an organization to be formed from a publicly available opinion. This "general opinion" [62] is derived from the experiences of a variety of different organizations with a certain organization, and facilitates the "imputation of a probability distribution over the various types" [56], or the trustworthiness, of an organization. Such a reputation can be subject to dynamism [103] in accordance with changes in the experiences of organizations, and the adjustment of the general opinion. The reputation provides organizations with information on which to form trust, without the requirement that such information is gathered on an individual basis by the organization. Such a reputation is analogous to branding which turns "experience goods" [104] such as Web services, whose properties cannot be inspected prior to usage, into "search goods" [104], whose properties can be inspected or inferred prior to usage, through the association of a certain reputation for trustworthiness with an organization. Moreover, the utilization of such information by organizations leads to reputation mechanisms acting as a "collaborative sanctioning systems" [64] and mechanism for "social control" [61]. Such "soft security mechanisms" [61, 64] provide "an answer to the inadequacy of traditional authorization mechanisms" [96], enabling the trustworthiness of an organizations to be defined over a continuous domain, rather than over the discrete domain of 'true' or 'false'. There are two classes of reputation mechanism [64]:

Centralized Reputation Mechanisms Centralized reputation mechanisms utilize a single source for information pertaining to the experiences of other organizations. They are commonly utilized in provision of Web services, with popular commercial examples including the feedback systems of Amazon [105] and eBay [106, 107] for further information.

Decentralized Reputation Mechanisms Decentralized reputation mechanisms utilize multiple independent sources for information pertaining to the experiences of other organizations. They are often associated with social networks, with a popular commercial example being the PageRank [108] system of Google [109] which derives the reputation of web sites based upon the hypertext linking structure of the WWW.

Reputation mechanisms can utilize a number of different computational techniques for the computation of reputation from the opinions of organizations. These techniques include [64]: bayesian systems, summation and averages, boolean, belief mechanisms, fuzzy mechanisms, flow mechanisms. The mechanism presented in [110] enables the computation of reputation for QoS, and incentivizes truthful opinions from organizations through payments based upon how close organizations are to the general opinion. Such a technique provides a means of subjective evaluation when objective evaluation is problematic. REGRET [111] is an example of a reputation mechanism which exploits the social structure of organizations to provide a weighting for the information utilized to compute reputation. Another reputation mechanism is presented in [112] to enable selection of Web services based upon the rankings of organizations. The reputation system in [113] is utilized to match requesters to certain Web services based on their defined policy for trust in QoS based on reputation. This system encapsulates the decision-making of the organization with regard to reputation in addition to the computation of the reputation.

2.5.6 Contract Mechanisms

Contract mechanisms enable trust to be formed in an organization through the negotiation and settlement of a contract with that organization. A contract refers to a "reliable promise by both parties, in which the obligations for each, are specified" [114]. The stipulation of such obligations provides a "commitment to reduce uncertainty" [71], with obligations verified and enforced by a court of law. This can transform SLAs into "legally binding contracts that establish bounds on various QoS aspects" [25], whose obligations are verified and enforced. The validity of the contract is dependent on the verifiability of obligations defined within the contract, with an obligation being verifiable if the "value is observable and can be proved before a court of law" [114]. This notion of verifiability for QoS is investigated in [115], and must deal with the presence of failures and delays, and absence of global synchronized time in a distributed system such as the Internet. There are two stages in a contract mechanisms:

Negotiation The negotiation stage involves the formulation of the obligations within the contract as a process of proposal and counter-proposal between organizations. Algorithmic Mechanism Design [116, 117, 118, 119, 120] studies the formation of mechanisms, or contracts, which incentivize certain outcomes from organizations. The notion of preference over outcomes is encapsulated within a "social choice function" [121] whose value must be optimized by a contract. Such mechanisms are designed to be "robust and non-manipulatable" [119] such that, for organizations, it is "irrational to deviate" [122] and behave in a manner which is untrustworthy and inconsistent with the obligations of the contract. This includes the exploitation of any informational or situational advantage by an organization. There are a number of examples of such mechanisms including those based upon

auctions [123, 124, 125] or bilateral bargaining [126, 127, 128]. Contract Theory [114, 129, 130] and Incentive Theory [131] studies such mechanisms solely as bilateral, contingent contracts between organizations. Conventionally, contingent contracts or "compensation and bonus mechanisms" [118] are utilized to address uncertainty for an organization through the separation of monetary payment for a Web service into two terms, compensation and bonus. The compensation is independent on the QoS, where as the bonus is calculated according to experience QoS of the Web service. The different types of contingent contract are discussed in [132], including warranty contracts, time trials, money back guarantees and general contingent contracts. Some examples of pricing structures for such contingent contracts are presented in [54, 133, 134]. These pricing structures seek to incentivize fulfillment of obligations within a contract through the calculation of specific monetary payments based on value and cost assumptions.

Settlement The settlement stage involves the realization of the obligations defined in the contract by the organizations. This introduces an additional stage at which organizations can exploit an informational or situational advantage, choose not to fulfill the obligations in the contract and behave in an untrustworthy manner. A court of law is required to determine the fulfillment or violation of an obligation and enact the appropriate sanctions, based upon monetary payment or freedom. Additionally, a court of law may be required to arbitrate over a contract in which the determination of fulfillment or violation of obligations is troublesome. This problem arises from the creation of incomplete contracts in which all possible outcomes are not appropriately stipulated. The enforcement of an electronic contract requires the ability to model such contracts, and the fulfillment and violation of obligations within these contracts, at run-time, as described in [135, 136]. Exchange protocols [72, 137, 138] provide a mechanism for the settlement of simple bilateral contracts, consisting of one obligation for each organization. A TTP is utilized in such protocols to achieve a certain "strength of fairness" [72] with protocols guaranteeing fairness with/without further communication or cooperation from an organization and with/without the utilization of an external arbitrator [138]. Fair exchange protocols [72, 138] assume that the organizations can be malicious and provide "guarantees for a correctly behaving party that it cannot suffer any disadvantages no matter whether the other party behaves correctly or tries to cheat...executing the protocol faithfully is safe for both parties" [139]. Rational exchange protocols [139] or Safe exchange protocols [140] assume that organizations are rational, and provide "incentives so that rational parties have more reason to follow the protocol faithfully than to deviate" [139]. Therefore, fair exchange and rational exchange protocols utilize different general socio-cognitive models as their building blocks.

The formation of contracts relies on a pre-existing regulatory system. This regulatory system may "constrain the set of rules that can be imposed" [141] such that there does not exist "complete

freedom in designing rules" [141] in a contract. Certain rules may not be valid within a contract due to pre-existing regulations. The legal implications of electronic contracts are the focus of the Uniform Electronic Transactions Act (UETA) [142] in the United States. This legislation deals with legal issues relating to electronic commerce and describes the formation of contracts between agents in the absence of human involvement [142, 143]:

> (a) operations of one or more electronic agents which confirm the existence of a contract of indicate agreement, form a contract even if no individual was aware of or reviewed the actions or results. (b) in automated negotiation, the following rules apply: (1) A contract may be formed by the interaction of electronic agents. A contract is formed if the interaction results in electronic agents engaging in operations that confirm the existence of a contract or indicate agreement. The terms of the contract are determined under section 2B-209(b). (2) A contract may be formed by the interaction of an individual and an electronic agent.

2.6 Management

Management involves the "monitoring, controlling, and reporting of, service qualities and service usage" [28] by an organization such that a Web service "fulfills the requirements of both owners and users of the system" [20]. A Web service is provisioned on a computing and communications infrastructure whose resource capacity, including computational, storage and network resources, is fixed at a certain point in time, and the capacity is partitioned across concurrent requesters of the Web service at any point in time. The resource capacity allocated to each concurrent requester directly affects the QoS experienced by that requester. For example, a requester which is allocated more computational and network resources will experience a higher QoS than a requester allocated less resources, *ceteris paribus*. In the presence of fluctuations in usage patterns of the Web service from requesters, the resource capacity which can be allocated to each requester is subject to the same fluctuations. This introduces difficulties for organizations in the management of their resources "when the workload they see is unpredictable" [25]. Additionally, resources in the infrastructure can be subject to byzantine and crash failures [144], which can remove their capacity for an indefinite period of time, and affects the QoS of all requesters over this period of time. The presence of uncertainty in the QoS arises from this variation in resource capacity, and the allocation of this capacity amongst concurrent requesters of the Web service. The management of a Web service seeks to address the allocation of resources such that a Web service can be provisioned to requesters with a certain QoS.

The management policy of a Web service defines a "goal, course or method of action to guide and determine present and future decisions" [145] such that a Web services is provisioned with a

certain QoS. Such policies enable an organization to more readily assure the QoS of a Web service. The organization can perform introspection on the value of system metrics, which reflect the state of the computing and communications infrastructure, and construct a model of the computing infrastructure at a given point in time, such that future QoS can be predicated based upon the model. The resource capacity of the infrastructure can then be controlled in accordance with these predictions and the management policy, in order to assure a certain QoS to requesters. This process of introspection and control in accordance with the management policy is encapsulated within a management control loop [20]:

1. Monitor the computing infrastructure to obtain a model of the computing infrastructure.
2. Interpret a management policy in order to make decisions about what controls must be enacted given the state of the computing infrastructure.
3. Enact the controls on the computing infrastructure to implement the decisions of the policy.

The management of a Web service can be based upon the fulfillment of obligations defined within an SLA. The management policy of the Web service can define controls such that these obligations are fulfilled by the Web service. This utilization of SLAs as a principal component in the management of a Web service is discussed in [38], with a comprehensive example of a software architecture for automated management of based on SLAs defined in [146]. The key stages in the management of a Web service based upon SLAs can be summarized as [40]:

1. Negotiation and specification of QoS requirements by the requester.
2. Establishment of SLAs between the requester and provider.
3. Mapping of QoS requirements to resource capabilities by the provider.
4. Reservation and allocation of resources by the provider.
5. Monitor parameters associated with QoS by the provider.
6. Adaption of the QoS to varying resource capacity by the provider.

A number of different management architectures [147] can be utilized to perform introspection on, and control of, the computing and communications infrastructure. These architectures define the informational and operational models [148] utilized to manage the resources in the infrastructure, and the granularity at which these management can be performed. Examples include Simple Network Management Protocol (SNMP) [149], Web Services Distributed Management (WSDM) [150] and Web-Based Enterprise Management (WBEM) [151]. The management architecture utilized is dependent on the nature of the infrastructure on which the service is provisioned, and the granularity at which management needs to be performed on the resources in this infrastructure.

2.6.1 Monitoring Mechanisms

Monitoring mechanisms refer to processes of "dynamic collection, interpretation and presentation of information" [20] which pertains to "various types of service usage and service quality" [28]. These mechanisms provide a basis for models providing introspection on the state of the computing and communications infrastructure, and the ability of the Web service to fulfill a certain QoS. The information utilized to define a model of state of the infrastructure can include system metrics such as queue length and availability. The types of model created by such mechanisms include [22]:

Historical Models Historical models extrapolate the QoS of historical service usage to predict the QoS of future service usage.

Observation Models Observation models estimate QoS of future service usage based upon the current model of the computing infrastructure.

Predictive Models Predictive models establish relations between the QoS of historical service usage, and historical observations of the state of the infrastructure to predict the QoS of future service usage.

The derivation of such models requires the gathering of appropriate information from the infrastructure. This information gathering can be utilize two different methods of information acquisition [20]:

Time-Driven Time-driven monitoring acquires information from the infrastructure at certain time intervals to derive a static view of the computing infrastructure.

Event-Driven Event-driven monitoring acquires information from the infrastructure on the occurrence of certain events to derive a dynamic view of the computing infrastructure.

In the following subsections, we introduce different management mechanisms which can be enacted by a provider, based upon the information provided by the monitoring mechanism, to control the QoS.

2.6.2 Admission Control Mechanisms

Admission control mechanisms [25, 152, 153] are "responsible for comparing the resource requirement arising from QoS levels associated with a new activity with the available resources in the system" [20]. Such mechanisms restrict concurrent service usage by requesters to a certain number

of admissions, such that this number remains within certain bounds for which the computing and communications infrastructure is able to fulfill a certain QoS. These mechanisms require an ability to attribute a certain resource capacity to each requester, with admissions based upon whether or not the appropriate resource capacity can be attributed to the next requester, such that the QoS remains within certain bounds.

2.6.3 Workflow Management Mechanisms

Workflow Management [146, 154, 155] or Reservation [156, 157, 158] mechanisms determine the order in which the interactions of different requesters are processed by the computing and communications infrastructure of the Web service. This ordering directly affects the QoS for a certain requester, specifically with regard to performance metrics. These mechanisms schedule service usage to decide those resources in the infrastructure to which interaction will be assigned and when these interactions will be assigned to resources. Workflow Management mechanisms pertain to Web services composed of two or more upstream Web services, such that the QoS arises from the composed QoS of the upstream Web services within the composed service. Such mechanisms can perform "global scheduling" [155] to map and manage the usage of upstream Web services in order to fulfill a certain QoS. Reservation mechanisms are similar to Workflow Management mechanisms, but pertain to non-composed Web services, where interactions are performed on resources within an infrastructure which resides in the control of a certain organization. Such mechanisms perform "local scheduling" [155] which assign the interactions of requesters to resources in the computing infrastructure in order to fulfill a certain QoS.

2.6.4 Provisioning Mechanisms

Provisioning mechanisms manage the physical resource capacity in the computing infrastructure in order to "allocate sufficient resources" [146] to a Web service and fulfill a certain QoS for the Web service. There are three different types of provisioning mechanism [146]:

Dedicated Provisioning Dedicated provisioning involves the utilization of a static set of resources within the computing infrastructure, for a static set of requesters and a static QoS. This is the simplest approach, involving the over-provisioning of resources, such that a certain QoS can be provided to a specific set of requesters regardless of the concurrent service usage and the state of the computing infrastructure. This over-provisioning is extremely inefficient for the provider, with the incurrence of high fixed costs to provide resources for periods of service usage which rarely occur. This result in resources remaining idle for long time periods, affecting the business metrics of the provider.

Per-SLA Virtual Provisioning Per-SLA virtual provisioning involves the support of SLAs defining a different QoS for different requesters, based upon the available capacity in the computing infrastructure at a given point in time. Per-SLA provisioning enables the available resource capacity of the computing infrastructure to be utilized to provide differentiated QoS, such that capacity can be efficiently utilized. The computing infrastructure may support a number of different SLAs at a given point in time in accordance with the state of the infrastructure. The QoS in a SLA is manipulated based upon the state of the infrastructure at a certain point i time, and can facilitate a reduction of the uncertainty in QoS through the statement of a QoS with greater expectation. Certain aspects of the computing infrastructure may reside outside the control of the provider such as dependencies on upstream services. The SLA may be adapted to changes in the state of the computing infrastructure, when further control of the resources in the computing infrastructure can no longer be performed. Such a mechanism is presented in [159] which considers the uncertainty in QoS experienced by the provider in the infrastructure, and enables the provider to offer a SLA based on the probability of fulfilling the stated QoS.

Dynamic Provisioning Dynamic provisioning involve the allocation and de-allocation of resources to and from the computing infrastructure based upon the number of concurrent requesters, the state of the computing infrastructure, and the QoS which the Web service is to fulfill. This enables the infrastructure to adapt to increased service usage with increased resource capacity in order to retain a certain QoS. These mechanisms must consider the trade-off between the increased costs of incurred by increased resources in the computing infrastructure, and the consequential effect on QoS, in order to decide whether such provisioning is beneficial. Such mechanisms are presented in [160, 161], which switch resources of the computing infrastructure between different services in accordance with the number of concurrent requesters for that service to retain a certain QoS for requesters.

Chapter 3

An Institutional Framework For Trust In Service Provisioning Relationships

This chapter addresses the problem of uncertainty in QoS experienced by the requester of a Web service. The main contribution of the chapter is a framework which facilitates the holistic representation of the environment of an SPR, and the subjective perception of this environment by the requester. The framework formulates the controls on the willingness of the provider to provision a Web service with a certain QoS as institutions. These institutions are perceived by the requester in order to derive a notion of trust, and decide upon participation in an SPR based on organizational objectives. Such an approach differs from previous work, which makes implicit or explicit assumptions regarding the environment without explanation of the implications of such assumptions for trust, and often focuses on environments containing a single class of trust mechanism rather than environments as a holistic encapsulation of the context of an SPR. Additionally, in contrast to previous work, institutions are utilized to model the constraints in the environment of an SPR, such that institutions are perceived by an organization to analyze, evaluate, and decide upon participation in an SPR. The value of this approach is to provide a basis for the evaluation of an environment of an SPRs in terms of the absence or presence of trustworthiness and trust, and the mismatch between trust and trustworthiness. This enables environments to be appropriately designed or manipulated by organizations to increase both trustworthiness and trust.

The remainder of the chapter is structured as follows. Section 3.1 introduces the problem addressed in this chapter. Section 3.2 provides a set of requirements for the solution to the problem. Section 3.3 defines the main assumptions of the framework. Section 3.4 introduces the fundamental concept of institutions utilized by the framework. Section 3.5 presents a framework to address the requirements and provide a solution to the problem. Section 3.6 discusses the applicability of Game Theory to evaluate the trust of a requester within a certain environment. Finally, the chapter is briefly summarized and concluded in Section 3.7.

3.1 Introduction

Uncertainty in QoS can be experienced by the requester of a Web service. The requester can be uncertain of the willingness and ability of a provider to provision a Web service with a certain QoS. This uncertainty arises from a lack of complete control over, and complete information pertaining to, the Web service. The provider can lack the willingness and ability to provision the Web service with a certain QoS, or the requester may be unaware of this willingness and ability. Such uncertainty leaves the requester vulnerable to the provision of a Web service with a QoS which does not meet certain requirements. This uncertainty can endanger the economic viability of SPRs, and mitigate any increases in organizational efficiency for the requester. The requester may prefer not to participate in SPRs given the presence of such uncertainties, due to the negative effects on organizational objectives from usage of a Web service with a QoS which does not meet requirements. Accordingly, the economic viability of SPRs is predicated on the ability of the requester to address this uncertainty.

The requester can address uncertainty in the QoS of a Web service with trust. Trust is a subjective belief in the trustworthiness of a Web service, and therefore its competence "to act dependably, securely and reliably" [51]. This belief can be formed from the utilization of a variety of different mechanisms, as discussed in Chapter 2. These mechanisms simultaneously reduce the uncertainty and increase the trust of the requester in the Web service through an ability to control the Web service, and to provide information pertaining to the Web service. Such mechanisms incur costs, which are attributable to the opportunity cost of resources which are consumed in the reduction of uncertainty. The extent to which costs will be tolerated to reduce uncertainty is based on disposition of the requester. The disposition of the requester determines the uncertainty in the QoS which is tolerable in an SPR given the control and information provided by trust mechanisms.

Trust is a contextual notion, such that changes in context lead to consequential changes in the trust of the requester. The context or environment of an SPR encapsulates all the controls to which the QoS of a Web service is subject, and the perception includes the perception of these controls by the requester in order to derive a notion of trust. Such controls may arise from technological, social or economic constraints, and there constraints can be highly interdependent. For example, the communication network on which organizations communicate may introduce technological constraints, such as unreliability or latency in communication, which affect the ability to impose economic constraints on organizations such as contract mechanisms. The ability of the requester to decide upon the economic viability of an SPR is subject to the ability to perceive these controls in a holistic manner, and to decide upon participation based upon organizational objectives. There exists a requirement to represent the environment, and the perception of the environment by the requester, such that the absence or presence of trust in an environment can be analyzed, and appropriately designed or manipulated to increase trust. Accordingly, this chapter presents a framework which

formulates the controls on the willingness of the provider to provision a Web service with a certain QoS as institutions, which are perceived by the requester in order to derive a notion of trust.

3.2 Requirements

The requirements define the aspects of the problem which must be addressed, and are defined as follows:

1. **A characterization of the environment of an SPR**
   The framework must characterize the environment of an SPR in a holistic manner, identifying the different controls which affect the QoS, and therefore trustworthiness, of a Web service.

2. **A characterization of the perception by the requester of the environment of an SPR**
   The framework must characterize the perception of the environment of an SPR, including the process by which trust is derived from perception and utilized to decide upon the economic viability of an SPR given organizational objectives and attitudes.

3.3 Framework Assumptions

The framework presented in this chapter to address the requirements is based upon a set of assumptions, which are defined as follows:

1. **Organizations are bounded-rational agents**
   The organizations which utilize and provision a Web service are assumed to be bounded-rational, and pursue their organizational objectives in a self-interested manner, subject to bounds on their computational, storage and communication resources. This assumption provides a basis for the cognitive processes of organizations, and enables organizations to be modeled economic actors, whose computational encoding is a rational agent. Such an assumption is sufficiently general to represent a large number of organizations, whose organizational objectives are unilateral revenue or profit maximization, subject to resource bounds.

2. **One or more third party organizations are mutually trusted by the requester and provider**
   The requester and provider of the Web services are assumed to mutually trust one or more third party organizations for the provision of certain functionality. These Trusted Third Party (TTP) organizations are able to provide Web services to the requester and provider to reduce uncertainty in QoS and support the economic viability of an SPR. Such an assumption

enables the framework to omit consideration of the manner in which trust in such a third party is formed, and to assume such organizations provide a root of trust. This avoids the complex recursive analysis which emerges when such organizations are assumed to rational, and themselves require incentives for the provision of their functionality.

3.4 Framework Fundamentals

The framework presented in this chapter to address the requirements defined in Section 3.2 utilizes the notion of institutions as the principal unit of control within the environment with which an organization interacts. Institutions represent sets of "formal and informal constraints" [162] defined over organizations to "limit the set of choices" [162] over their actions. These constraints "regulate the interactions" [163] of organizations and facilitate "collective action in control, liberation and expression of individual actions" [164]. For example, the rules of law constrain the sets of individual actions in a society in order to expand the set of collective actions. This can involve constraints which "compromise between competing objectives" [165] of organizations in order to provide "stable expectations of behavior" [166] such that collective action is viable in a society. Organizations in a society may have a combination of mutual and conflicting objectives over certain actions. Institutions can provide constraints on individual actions where the objectives of organization are in conflict, in order to realize actions where the organizations have a mutual objective. The set of institutions which define constraints over a specific action to be taken by an organization can be referred to as the "context" [51], "transaction context" [166] or "social context" [167] of that action.

Institutions can be viewed as attaining coordination amongst organizations in a society. Coordination is the mutual selection of behavior by organizations such that a certain outcome is realized. This mutual selection imposes order and can eliminate "bottlenecks and disjunctures" [168] the mutual selection of actions by organizations. The coordination of actions is often realized through the cooperation of organizations. Cooperation is the adjustment of the objectives of an organization to the real or anticipated objectives of another organization [169]. This is often presented in contrast to the notion of competition amongst organizations, in which coordination is obtained through a conflict in objectives. In reality, cooperation is required to some degree in competition, otherwise competition would be mutually destructive amongst organizations. The outcome on which the actions of organizations coordinate can vary in the extent to which it pursues the individual objectives of the organizations, in terms of organizational efficiency, and the collective objectives of society, in terms of aspects such as fairness. Often, the individual objectives of organizations and the collective objectives of society are balanced in some manner to provide compromise between efficiency and equity.

Figure 3.1 illustrates the different dimensions of cooperation as defined in [169]. These dimensions

Name	Description
Mutuality of Interests	The inherent alignment of the objectives of organizations.
Shadow of the Future	The adjustment of objectives due to potential retaliation or damage in the future for an action in the present.
Sanctioning	The adjustment of objectives due to the ability to detect and sanction organizations for certain actions.

Figure 3.1: A Table Of The Dimensions Of Cooperation

refer to contextual aspects in the environment which determine the propensity of an organization to cooperate with another organization. The dimension of mutuality of objectives represents an inherent alignment of objectives between organizations over certain actions. The dimension of the shadow of the future represents an alignment of objectives which results from the potential retaliation or damage in the future. Such a dimension can attain cooperation when the objectives of organizations are inherently in conflict. The dimension of sanctioning represents an alignment of objectives which results from the ability to detect and sanction organizations for certain actions. In a similar manner to the shadow of the future, this dimension enable cooperation to be attained when the objectives of organizations are inherently in conflict.

The constraints defined by institutions require definition and enforcement. The definition of an institution enables its internalization by organizations such that the actions of the organization can reflect the constraints defined by that institution. This is particularly pertinent for agents which required an appropriate encoding of an institutions in order to internalize the constraints of that institution. The enforcement of institutions provide a "credible threat" [170] by which the adherence of organizations to the rules of the institution can be legitimized. Such a credible threat may involve "some agency outside the institutions that acts to provide necessary incentives" [171]. These agencies or organization define constraints in accordance with certain objectives, and enforce these constraints with appropriate threats. For example, criminal law is defined by the Government and is enforced by the Police and the Courts utilizing the credible threat of expropriation of property rights or freedom. The manner in which an institution is defined and enforced determines the "pillar" [172] on which the institution is supported. Figure 3.2 provides a summary of these different pillars, as defined in [172]. These pillars define institutions in different manners, and utilize different types of credible threat, such as legal sanctions, moral governance or cultural support, to enforce the constraints of the institution.

Institutions define constraints which are legitimate within a certain domain or society. These domains define constraints over the organizations or individuals in that domain, and enforce these

	Regulative	Normative	Cognitive
Basis Of Compliance	Expedience	Social Obligation	Taken For Granted
Mechanisms	Coercive	Normative	Mimetic
Logic	Instrumentality	Appropriateness	Orthodoxy
Indicators	Rules, Laws, Sanctions	Certification, Accreditation	Prevalence, Isomorphism
Basis Of Legitimacy	Legally Sanctioned	Morally Governed	Culturally Supported
Cultures	Rules, Laws	Values, Expectations	Categories, Typifications
Social Structures	Power Systems	Authority Systems	Identities
Routines	Protocols	Conformity	Scripts

Figure 3.2: A Table Of The Three Pillars Of Institutions

constraints through an appropriate credible threat. Those organizations which reside within the domain are subject to the constraints of the institutions. Examples of such domains include political domains such as the United Kingdom or organizational domains such as Newcastle University. An organization may reside within a number of domains and be subject to the constraints of the institutions in those domains when performing actions. These domains may be inter-dependent such that the constraints of one domain assume those of another domain on which it is dependent. For example, the institutions defined by Newcastle University assume those of the United Kingdom. The participation of an organization in domain may be voluntary, such that the organization can choose to be bound by such constraints, or mandatory, such that the organization is perpetually bound by such constraints. Figure 3.3 illustrates the notion of a domain which defines constraints over one or more organizations, defined by some organization A, and enforced by some organization B. For example, organization A may be the government and organization B may be a court of law.

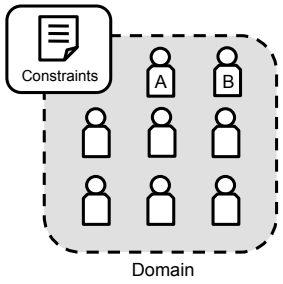

Figure 3.3: An Illustration Of Institutional Domains

Organizations are subject to bounds on computational, storage and communication resources, or technological constraints. The Web services which represent these organizations reflect such bounds. This can restrict those constraints which can be defined and enforced within a domain in which the organization resides. For example, a constraint may be defined, but due to technological constraints, such as the inability to monitor the interactions of organizations, may not be enforceable. This inability to effectively legitimize adherence to constraints leads to the redundancy of constraints and

the absence of the required control over organizations in a domain. Institutions and organizations co-evolve with changes in technological constraints, with the requirement for institutional change driving technological changes, and vice-versa. Accordingly, the design and manipulation of institutions must consider the technical constraints to which institutions in a domain are subject, as these technological constraints determine the efficacy with which constraints can be legitimized within a domain.

Institutions provide a generic approach to control over the actions of organizations. They have been considered in previous work relating to multi-agent systems such as [173, 174, 175], but an explicit formulation of trustworthiness and trust with regard to institutions has not been attempted in previous work. We believe that such an approach can provide a generic manner in which the context or environment of an organization, and of organizational decision making can be effectively represented and analyzed. This chapter provides such a formulation of the trustworthiness of an organization, and the trust placed in such an organization by other organizations. Trustworthiness is a manifestation of the institutions to which the actions of an organization are subject. This is defined by the domains in which the organization resides and the legitimacy of the institutions within these domains. Trust is determined by the perception of the constraints to which an organization is subject by other organizations. This perception may or may not reflect the actual institutions to which the organization is subject. The important difference between trustworthiness and trust is this notion of perception, which can lead to inconsistencies between the trustworthiness of an organization, and the trust placed in the organization by other organizations.

3.5 Framework Components

There are two principal components within the framework: environment and requester. The environment encapsulates the institutions and the organizations which define and enforce these institutions. The requester encapsulates the perception of the environment, and the reasoning process of an organization based upon this perception. In this section, we utilize the concept of institutions to provide a detailed description of the concepts of environment and requester in an SPR, and the relationship between these concepts with regard to trustworthiness and trust, utilizing the Example Scenario defined in Section 1.1.

3.5.1 Environment

The environment of an SPR is defined to comprise of three different layers: Service Layer, Platform Layer and Communication Layer. These three layers provide a holistic representation of the environment of an SPR. The layers each comprise of one or more organizations, or Web services, which define and enforce institutions which constrain QoS. The institutions at a given layer define constraints on which the layers above are dependent. For example, the constraints defined and en-

forced at the Platform Layer are dependent on the constraints which are defined and enforced at the Communication Layer. Figure 3.4 provides an overview of the environment and the layers within the environment.

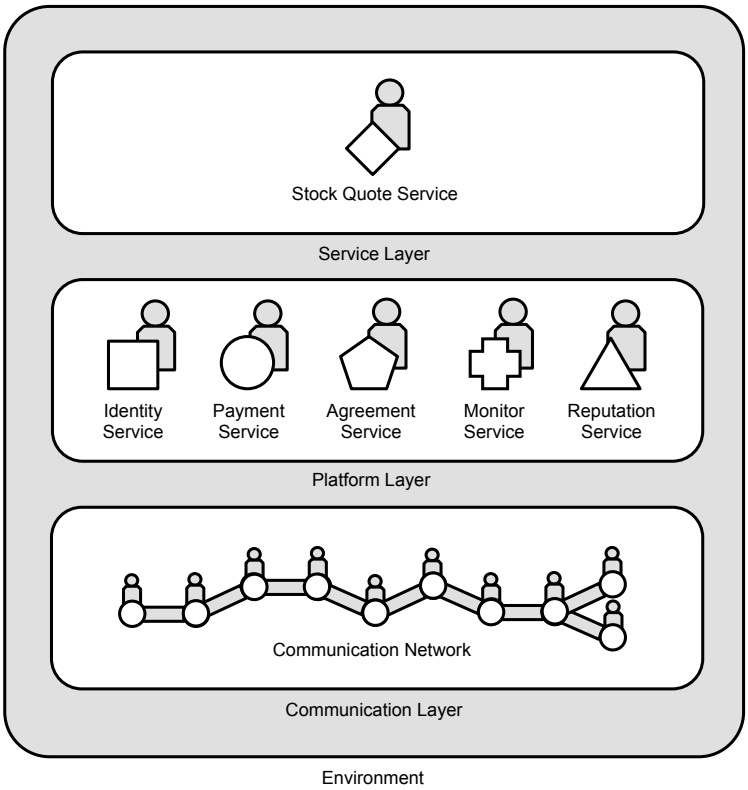

Figure 3.4: An Illustration Of The Environment Of An SPR

In the next three subsections, we provide a description of each of these layers, and the interdependencies amongst the layers.

3.5.1.1 Service Layer

The Service Layer comprises of the Web service for which the SPR is formed. For example, the Stock Quote Service of XYZ Stockbrokers. This Web service encapsulates a set of institutions which inherently constrain the QoS of the Web service. These institutions represents cognitive institutions, which define the inherent reasoning process to which the Web service adheres and which determine the QoS of the Web service. For example, the management policy of the Stock Quote Service is a cognitive institution which is defined and enforced by the XYZ Stockbrokers and determines the inherent QoS of the Web service. The bounds on computational, storage and network resources within the computing and communications infrastructure on which the Web service is provisioned, or bounded-rationality, of the provider dictate that this reasoning process may not be optimal with

regard to organizational objectives. Indeed, the process simply reflects the self-interested pursuit of organizational objectives subject to resource bounds, such that the process may be theoretically sub-optimal. The objectives of the organization can be defined within a utility function over QoS, $u_m(v)$, where m is a specific metric and v is an assigned value for that metric. The management costs of the organization can be defined with a cost function, $c_m(e)$, which expresses the cost to provision a Web service with a certain QoS, and reflects the consumption of scarce resources, or effort, e, in the management of the Web service.

The provider experiences uncertainty in the QoS, given the management of the Web service, due to factors outside his control, such as hardware and software failures or fluctuations in service usage. Therefore, QoS can be represented a non-deterministic function of the management costs, or effort, of the provider, with the provider uncertain of the QoS of the Web service given the exertion of a certain effort. This non-determinism is a reflection of the ability of the provider to effectively manage the computing and communications infrastructure on which the Web service is provisioned. Such uncertainty in QoS can be represented by a random variable, V_m, with an associated probability distribution, $f_m(e|v)$. The probability distribution imposes a real value on the QoS of a Web service, which represents the degree of belief held by the provider that the Web service will be provisioned with a certain QoS, (m, v). This probability distribution is conditional on the management of the Web service the provider, and can be derived by the provider from monitoring mechanisms such as those described in Chapter 5. It can be reasonably assumed that the greater the effort exerted the higher the probability of a better QoS. Equation 3.1 illustrates the expected utility function of the provider, which considers the uncertainty experienced by the provider in the QoS given management.

$$EU = \int_{v \in \mathbb{R}} f_m(e|v) \cdot u_m(v) - c(e) \tag{3.1}$$

The form of the utility function, $u_m(v)$, defines the manner in which utility varies with the QoS and the attitudes or disposition of the provider to accept uncertainty. The provider will participate in an SPR when $EU \geq \overline{b}$, where the real value, $\overline{b}$, represents the minimum utility required by the provider from participation in SPR, given the beliefs. For example, should $\overline{b} = 0$ then the provider requires that the expected utility from the SPR is positive. This is known as the individual rationality constraint in economic literature. More importantly, the provider will provide a certain QoS, (m, v^*) when the utility from the service from such a QoS is greater than the utility from any other QoS, (m, v). This is known as the incentive compatibility constraint in economic literature, and is the basis for verifying that there exist sufficient incentives for a provider to realize a certain QoS. Under the assumption of a positive cost for management of the Web service, there must exist an incentive for the provider to provide the Web service with a certain QoS and to participate in the SPR. The participation of the provider must yield positive utility over and above the cost of management,

such that the individual rationality constraint is fulfilled, and must yield utility over and above cost for a certain QoS than for lower QoS, such that the incentive compatibility constraint is fulfilled. Accordingly, there must exist incentives in the environment of the SPR which can incentivize the provider, not only to participate, but to do so with a certain QoS. The trustworthiness of the Web service, and the willingness and ability of the Web service to exhibit a certain QoS is predicated on the existence of such incentives in the environment.

In the next section, we describe the Platform Layer which introduces incentives to the environment in order to incentive a certain QoS.

3.5.1.2 Platform Layer

The Platform Layer comprises of the Web services which define and enforce institutions to manipulate the willingness of a provider to provision a Web service with a certain QoS. In order to manipulate the willingness of the provider the cognitive process by which the provider manages the Web service must be known. This enable the institutions provide appropriate incentives to manipulate the utility function of the provider, and fulfill the individual rationality and incentive compatibility constraints of the provider, such that the Web service is provisioned with a certain QoS. This manipulation of the utility function constitutes the cooperation of the provider with the requester, with the adjustment of objectives arising from the the presence of appropriate situational dimensions such as sanctioning and the shadow of the future, as defined in Section 3.4. It can be reasonably assumed that the utility function of the provider expresses objectives over monetary payments, and that the provider prefers larger monetary payments within an SPR than smaller monetary payments. Therefore, the incentives defined by an institution must be such that the exertion of more effort in the provision of Web service, or incurrence of larger management costs, is rewarded with an expectation of greater monetary payments. The institutions which manipulate the willingness of the provider can be supported by two different pillars, as defined in Section 3.4:

Normative Institutions Normative institutions provide control through social obligation, and an effect on the future ability of the provider to participate in SPRs with requesters and yield further monetary payments. For example, a provider with a bad reputation arising from the social sanctions of a normative institution may struggle to participate in SPRs in the future.

Regulative Institutions Regulative institutions provide control through sanctioning, and an effect on the monetary payments of the provider from participation in an SPRs. For example, a provider who incurs sanctions will yield smaller monetary payments from a certain SPR than a provider who avoids such sanctions.

The set of Web services which can be utilized to provide definition and enforcement of normative and regulative institutions at the Platform Layer are described below:

Identity Service The Identity Service encapsulates the functionality to perform authorization and authentication of organizations within a domain. This enables the actions of organizations to be associated with a persistent identity, and provides a basis on which organizations can make binding commitments to actions, and be held accountable for the fulfillment of violation of these commitments. Such accountability is the basis of the sanctions in a domain which incentivize certain actions, such as the provision of the Web service with a certain QoS. In the absence of a persistent identity, organizations are anonymous and incur no future consequences from their actions within the domain. Therefore, the organizations in the domain cannot be distinguished, and those organizations which exert appropriate effort to provision a Web service with a certain QoS are indistinguishable from those which exert no effort and seek to exploit requesters. The service requires power over the creation of identities by organizations, and the authority to associate an organization with a specific identity. Such services are discussed in detail in [23, 176].

Payment Service The Payment Service encapsulates the functionality to transfer monetary payments between organizations in a domain. These monetary payments act as a common medium of exchange and store of value, such that organizations need not engage in the direct exchange of Web services, or barter, which relies upon a 'double coincidence of wants' amongst the organizations. The organizations are assumed to ultimately express objectives over such payments. The provider can be compensated by the requester for the exertion of costly effort in the provisioning of a Web service with a certain QoS through an appropriate payment. Additionally, the provider can be sanctioned for the exertion of no effort in the provisioning of a Web service, through an appropriate payment. The assumption that the provider expresses objectives over monetary payments, means that the provider is indifferent between the costly effort to provision the Web service with a certain QoS and a certain payment. Therefore, there always exists some payment which is equivalent in terms of utility to the cost of the effort to provision the Web service with a certain QoS, meaning that the provider can always be incentivized by an appropriate monetary payment. In order to facilitate the transfer of monetary payments, the service require sufficient power over the monetary resources of organizations such that the service can perform the required expropriation and appropriation of these resources. Such services are discussed in detail in [177, 178] with an example of such a service included in [179].

Agreement Service The Agreement Service encapsulates the functionality to explicitly define the obligations of organizations within a domain, such that the actions of organizations are constrained

by such obligations. These obligations can include the provision of the Web service with a certain QoS, and the transfer of monetary payments between organizations. The monetary payments defined can be contingent on the provision of the Web service with a certain QoS or can be non-contingent on the QoS. Commonly, the obligations are explicitly defined within an SLA produced by the Web service. The obligations within the SLA can be subject to manipulation by organizations in a process of negotiation which reflects the objectives of each organization, before both organizations commit to the obligations within a certain instantiation of the SLA. The Agreement Service requires the authority to bind the organizations to this instantiation of the SLA utilizing their identities such that each organization can be held accountable for the fulfillment or violation of these obligations. Such obligations must be designed in a manner which provides sufficient incentives for the provider to provision the Web service with a certain QoS, and therefore satisfy his individual rationality and incentive compatibility constraints. This design is the focus of Incentive Theory and Contract Theory in economics, with examples of their application to QoS in SPRs including [110, 134]. The design of such SLAs is based upon the pillar on which the institution represented by the SLA is supported. In the case of a normative institution, such SLAs may define a QoS and monetary payment to the provider, $((m, v), p)$, for the provision of the Web service which is not contingent on the QoS experienced by the requester. The incentive of the provider to provision the Web service with the QoS is based upon social obligation and the shadow of the future, with a lack of costly effort and appropriate management affecting his future ability to participate in beneficial SPRs. In the case of a regulative institution, such SLAs may define one or more QoS levels and one or more monetary payments to the provider, $((m, v_1), p_1), ((m, v_2), p_2), \ldots, ((m, v_n), p_n)$, for the provision of the Web service which are contingent on the QoS experienced by the requester. The incentive of the provider to provision the Web service with the QoS is based upon monetary payments and sanctions, with a lack of costly effort and appropriate management affecting the monetary payments yielded from the SPR. The adherence of organizations to the obligations defined in an SLA, whether as a normative or regulative institution, is predicated on the existence of an appropriate credible threat which enforces the institution. Such enforcement requires the ability to detect the fulfillment or violation of obligations, and to enact sanctions based upon such fulfillment or violation. In the absence of such enforcement, the SLA is useless as a device for constraining the provider to provision a Web service with a certain QoS, and therefore for increased trustworthiness.

Monitor Service The Monitor Service encapsulates the functionality to detect the QoS, (m, v), of a Web service in a domain. This provides the verification of obligations defined within an SLA, whether normative or regulative, such that the institution can be enforced. This enforcement involves the enactment the appropriate sanctions based upon the provision of the Web service with a certain QoS, such as social or monetary sanctions. The Monitor Service is coordinates the detection and

sanctioning of obligations within the SPR, such that beyond request to the Web service by the requester, the monitor performs all other interactions within the SPR, and defining the sequencing constraints on Web service invocations at the Platform Layer and the Service Layer. For example, the Monitor Service will check the value of the response time metric to ensure that the QoS obligation of 200 milliseconds is fulfilled by the provider, and on violation invoke the Payment Service to perform a transfer between the requester and provider. The organization which provisions the Monitor Service must have sufficient authority within a domain to enact such sanctions. The degree of authority is based upon the organization in the domain which provisions the Web service, such that a TTP which is deemed trustworthy by many organizations has a higher degree of authority in the provision of the Monitor Service than the requester or provider. For example, a Monitor Service which is provisioned by the requester is unlikely to have sufficient authority to enact payment sanctions on the provider, but may have sufficient authority to enact social sanctions on the provider through reputation.

Reputation Service The Reputation Service encapsulates the functionality to provide a public opinion of a provider within a domain. This creates a credible threat to the provider to enforce a normative institution by persistently associating the QoS of a Web service in previous SPRs with that Web service, or creating a 'shadow of the future'. The service facilitates the association of reputation effect with a QoS which can be quantified by the profit which is foregone from the effect of a certain QoS on future SPRs. This effect is determined by the efficacy of the Reputation Service in punishing a certain QoS by the provider. Such efficacy relies upon the authority of the Reputation Service within a domain; the greater the authority of the Reputation Service, the more credible the threat against the provider for a certain QoS. For example, if there are only a small number of requesters which utilize the Reputation Service, it may not have sufficient authority to impose any control on the behavior of the provider through social obligation. The authority is predicated on the existence of the persistent identity provided by the Identity Service, such that the fulfillment of obligations by an organization can be persistently associated with an identity, and utilized not only to provide information to organizations to initialize beliefs (see Section 3.5.2.1) but also to control the organization and provide a credible threat.

In the next section, we describe the Communication Layer which introduces constraints on the communication between Web services at the Platform Layer and Service Layer and influences the ability to constrain the QoS at these layers.

3.5.1.3 Communication Layer

The Communication Layer comprises of the services which constrain the communication between organizations in an SPR. These constraints are encapsulated within communication protocols, which

define the syntax, semantics and ordering of communication between organizations. The constraints defined by the communication protocols at the Communication Layer affect the properties of communication between the Web services defined in the Platform Layer and Service Layer. These communication protocols exhibit certain properties based upon the communication channels on which the messages are communicated. Such properties pertain to qualitative aspects of communication such as reliability and latency. It is these properties which directly affect the communication at the Platform Layer and Service Layer. For example, the failure in a communication channel can result in the failure of a request to a Web service, and a resultant effect on QoS of that Web service. These qualitative aspects can emerge from the self-interest of the organizations which provision the communication services such as ISPs. The communication network at the Communication Layer can contain sets of communication channels which are provisioned by a number of different organizations. These organizations can manage communication channels in a manner which is consistent with their beliefs, attitudes and objectives. For example, a malicious organization may provide a communication service which corrupts or loses messages. This can lead to the actions at the layers above being subject to failures or variation in message delays due to the organizations which provision each communication channel.

The impact of the Communication Layer on the QoS of the Web service at the Service Layer provides difficulty for the provider in provisioning the Web service with a certain QoS. There are factors in the Communication Layer which affect the QoS but reside outside the control of the provider. In the commitment to certain obligations for QoS, the provider is subject to the uncertainty which is introduced at the Communication Layer. The provider may not wish to adopt all this uncertainty due to the impact of its revenue due to factors outside of its control, and indeed may wish to share such uncertainty with the requester. This sharing may take the form of offering a range over which the value of a metric may fall, which considers the effect of the Communication Layer. Alternatively, the provider may constrain the QoS at the Communication Layer through participation in upstream SLAs with the different providers of the services at the Platform and Communication Layer in terms of QoS, such that these services have QoS obligations which must be fulfilled. This yields a potential deep and expensive chain of contracts which detect and sanction the performance of functionality by each service at each layer in terms of QoS. This is investigated in [180] with regard to self-interested message propagation by the routers of certain organizations.

Additionally, there exists no global view of state amongst the organizations at this layer, with different organizations perceiving a different QoS from measurement at different points in the communication network. This inability to derive a global notion of state for the QoS directly affect the ability of the Platform Layer to impose constraints on the QoS of the Web service at the Service Layer. The Monitor Service perceives the QoS of the Web service at the Service Layer subsequent to the effect of communication channel between this Web service and the Monitor Service. Ac-

cordingly, the properties of the communication channels which are utilized for an SPR have a large influence on their viability since the experience of the requester couples together the influences of both the provider and communication channel making QoS obligations troublesome to enforce in an objective manner. The requester is unaware that the lost response has arisen due to a failure of the communication channel rather than a failure of XYZ Stockbrokers itself. The requester may have to accept that the QoS obtained from the provider may at points fluctuate due to the communication channels rather than the willingness and ability of the provider of the primary Web service. Therefore, whilst the requester forms an SPR with the provider for a service with a certain QoS, this QoS can be affected by communication channels outside the control of either organization. Figure 3.5 illustrates the difference in the views of the organizations, where an outcome (m, v) is perceived by XYZ Stockbrokers, an outcome (m, v') is perceived by DEF Agreements, a third party organization, and an outcome (m, v'') is perceived by the ABC Investments (if ABC Investments provisions the Monitor Service then $v' = v''$).

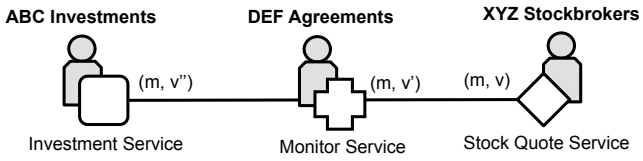

Figure 3.5: An Illustration Of The Communication Layer Effects

The control of the services at the Communication Layer, or indeed the Platform Layer, in order to receive a certain QoS can be analyzed in an analogous manner to the Web service at the Service Layer, by placing the Web service at the Service Layer of the framework and defining the Platform and Communication Layer in an appropriate manner to exert the control required, given the organization provisioning the service. We can reasonably assume that there will be services at the Communication Layer which are outside the control of either the requester or provider. Indeed, there will be aspects which are outside the control of the services at this level such as fluctuations in demand for the service (e.g., high load at peak times for Internet Service Providers (ISPs)). The services at this level will be unlikely to wish to enter into strict SLAs with a potential vast number of requesters for QoS, since it would be extremely challenging to manage and control to the extent that obligations will be regularly met. Accordingly, it is likely that the requester and provider will both have to accept some degree of uncertainty in the QoS themselves. SLAs due not provide complete control over the QoS obtained from a provider for the requester, and therefore cannot be complete trustworthy from an objective perspective, they may only be trustworthy to the extent that the requester is willing to participate from a subjective perspective, and accept the uncertainty at the Communication Layer. The requester and/or provider must accept the uncertainty introduced by the communication channel or face having to engage in an ever-increasing hierarchy of SLAs.

In the next section, we describe the perception of the environment by the requester in order to derive a notion of trust, and decide upon participation in an SPR based on organizational objectives.

3.5.2 Requester

The requester perceives the environment of an SPR in order to derive a notion of trust in the QoS of a Web service, and decide upon participation in the SPR based on organizational objectives. The trust in the QoS represents a subjective belief in the presence of institutions within the environment which constrain the QoS, or in the trustworthiness of the environment. Therefore, whilst the environment may be trustworthy, the subjective belief of the requester may be such that there does not exist a sufficient belief in this trustworthiness to participate in the SPR. The perception of the environment includes the perception of the three layers of the environment which define constraints on QoS. Accordingly, the perception of the environment includes:

- **Web Service at the Service Layer**
  This represents a perception of the willingness and ability of the Web service to provide a certain QoS.

- **Set of Web Services at the Platform Layer**
  This represents a perception of the willingness and ability of the Web services at the Platform Layer to provide incentives for a certain QoS from the Web Service at the Service Layer.

- **Communication Network at the Communication Layer**
  This represents a perception of the willingness and ability of the communication networks to enable interaction with the Web Services at the Service Layer and the Platform Layer.

The requester has beliefs with regard to the different Web services at the Service Layer and Platform Layer, and communication network at the Communication Layer. These different beliefs enable a model to be created for a specific environment in which an SPR may be formed. For example, ABC Investments may have beliefs relating to the willingness and ability of XYZ Stockbrokers to provision the Stock Quote Service with a certain QoS. In the event of a change in the environment, such as a change in the Communication Network or one or more Web Services at the Platform Layer, a change in the model of the environment is required. The model represents the beliefs that the requester has, not only that the incentives defined at the Platform Layer are sufficient to incentivize the provider, given his rationality, but also that the Web Services at the Platform Layer can enforce the institution to provide these incentives. The beliefs of the requester over these factors of the environment are formed from information gathering and processing. Such information can be obtained from a variety of different information mechanisms such as first-hand monitoring

or reputation mechanisms. This information may also represent encoded human information from the organization on whose behalf the requester interacts participates in SPRs. In essence, the information pertains to the institutions in the environment which constrain the QoS. The presence of incomplete or imperfect information leads to models of the environment which do not reflect the trustworthiness of the environment, and leads to an absence of trust when trust would be justified under complete and perfect information, or the presence of trust when trust is not justified under complete and perfect information. Figure 3.6 illustrates the perception of the environment by ABC Investments.

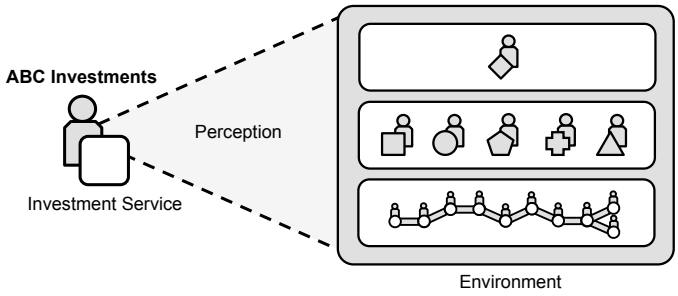

Figure 3.6: An Illustration Of Perception Of The Environment By The Requester

These information mechanisms through their construction of models affect the perception of requesters and can therefore provide a credible threat which controls behavior. So, information mechanisms provide information on the control mechanisms present in the environment, and in doing so provide a model for the requester of the environment. Information from extrinsic mechanisms can act as control on behavior of provider as it knows that these will be used by providers to decide on participation in SPRs and so may prevent them from acting in a certain manner for fear of the shadow of the future. These beliefs can be refined over time by the requester utilizing learning models such that the beliefs reflect the latest information. Indeed, the architecture used to realize the requester, such as the utility-based agent architecture shown in Chapter 2 defines the manner in which the beliefs of the requester are created, updated and incorporated into the decision-making of the requester. Accordingly, the agent architecture determines the optimality of the decision-making by the requester, with heuristics often utilized to provide good, approximate decisions under the inherent computational constraints of the agent implementation. We do not explicitly consider the information mechanisms from which such beliefs may be formed simply that such beliefs exist and represent the state of information at the point of decision by the requester.

The perception of the environment includes the Web services at the Platform Layer which are provisioned by a TTP. The organization which provisions the Web service determines the willingness and ability of the service to provision the necessary functionality to realize the incentives. A Web service provisioned by the TTP is assumed to be trusted by the requester, such that there exists no

uncertainty over the functional or non-functional properties of the Web service. The organization requires no incentives beyond a single payment to perform this functionality. Such payments reflect a class of transaction costs incurred by the requester in order to reduce uncertainty in the QoS. The TTP can therefore be considered willing and able to provide a certain Web service for a certain payment and will not act strategically in the provision of the Web service. For example, the TTP will not avoid costly effort in order to receive the payment without cost. Accordingly, the TTP is consider to be inherently honest, rather than rational. In the event that a third party is consider to be rational, appropriate incentives must be provided for the TTP to provision the functionality to support the SPR at the Platform Layer. This yields a recursive analysis of Web services at the Platform Layer in the same manner as with the Web service at the Service Layer, such that the provision of incentives for these services can be analyzed in an analogous manner to the incentives for the provider of the Web services at the Service Layer. This yields a recursive analysis of the Web services at the Platform Layer at the Platform Layer until there exists at some level of recursion only Web services at the Platform Layer which are provisioned by benevolent organizations. These organizations can be considered the root of trust in any environment of an SPR, providing a Web service whose functionality does not need to be incentivized. The requester may have beliefs with regard to the Web services provisioned by certain TTPs, such that their services are always trusted. For example, the requester may have beliefs that a service provided by the government is completely trusted. The Web services for which the requester has beliefs are conventionally dependent on the administrative, social, geographical and political domains in which the requester resides. This constrains the set of potential services which can be utilized to support an SPR, with it often necessary for certain Web services to trusted by both the requester and provider in the SPR. This dictates that a certain degree of commonality must exist in the beliefs of the requester and provider for an environment to be feasible for a certain SPR.

Figure 3.7 provides a simple illustration of these different domains, where domain A could represent the political domain of a single government, domain B could represent the administrative domain of a single organization, domain C could represent the administrative domain of a trade body, and domain D could represent the social domain of a friendship group. The constraints defined within domain B and domain C are dependent on the constraints defined in domain A. The labels R, M, P, I, A represent the organizations which provision the Reputation, Monitor, Payment, Identity and Agreement Services within each domain. There may exist zero or more of these services within the domain based upon the social relations in the domain. The domains contain the social relations between organizations such as those of authority and power. Such social relations are vital for the control of the provider through Web services provisioned by TTPs at the Platform Layer. For example, in domain B there will exist relations of power between the government and the requesters and providers in the domain. Each domain can have certain Web services which are trustworthy for

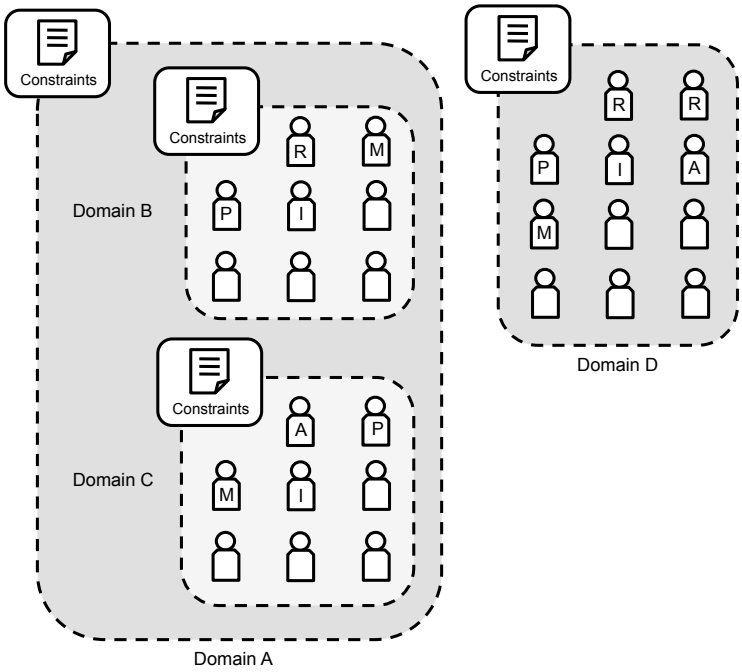

Figure 3.7: An Illustration Of The Domains At The Platform Layer

requesters in that domain. For example, a government may provide a Web service which is trustworthy for requesters in a certain political domain, but not trusted in other political domains. Such issues may be mitigated through the utilization of trusted modules within untrusted organizations rather than untrusted modules in trusted organizations to provide functionality. One could envisage that the requester would more easily trust a module whose functionality can be verified, than an organization whose objectives and functionality may not be verifiable.

3.5.2.1 Participation

The rationality of the requester dictates that participation in an SPR reflects the unilateral pursuit of organizational objectives. In accordance with this model, the objectives of the requester are defined within a utility function, $u_m(v)$, which expresses objectives over the QoS, (m, v) for a given metric, m and an assigned value for that metric, $v \in V$. The utility function contains a value function, $w_m(v)$, which expresses the value to the requester of the Web service with a certain QoS, (m, v), and a payment $p_m(v)$ for that QoS.

$$u_m(v) = w_m(v) - p_m(v) \tag{3.2}$$

We consider two distinct outcomes of an SPR from the perspective of the requester, the fulfillment of a certain QoS or the violation of that QoS. The value function can be represented by a step function to reflect the significant difference in value between these outcomes for the requester. Figure 3.8

illustrates a step function for ABC Investments, which requires that the response time of the Stock Quote Service is less than 200 milliseconds, and on violation of this bound the value of the Web service diminishes significantly towards zero. The QoS experienced by the requester is subject to uncertainty, given that the requester cannot verify the QoS which will be experienced from a Web service prior to participation in an SPR. This uncertainty in QoS for a certain environment is reflected in the beliefs of the requester which can be represented by a random variable[1], V_m, with an associated probability distribution, $f_m(v)$, which imposes a real value the QoS of a Web service representing the degree of belief that the Web service will exhibit those properties. This probability distribution emerges from the model of the environment held by the requester. The requester requires a method by which an encoding of the environment, and the institutions within that environment, as perceived by the requester, can be evaluated to yield such a probability distribution. This will involve the appropriate consideration of dependencies amongst institutions, and the constraints defined within these institutions. The detail of such an encoding and evaluation is omitted from consideration in this chapter, and forms part of the future work which arises from the thesis.

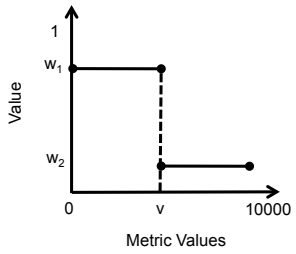

Figure 3.8: An Illustration Of A Step Function

The objectives and beliefs of the requester can be combined to yield an expected utility function from an SPR:

$$EU = \int_{v \in \mathbb{R}} f_m(v) \cdot u_m(v) \tag{3.3}$$

The intentions of the requester are determined by the form of the utility function, $u_m(v)$, in conjunction with a real value, $\overline{a}$, representing the minimum utility required by the requester from an SPR, given the beliefs. The form of the utility function defines the manner in which utility varies with the QoS and the disposition of the requester to accept uncertainty and participate in an SPR. This disposition is analogous to the notion of risk-propensity in economic literature, where organizations can be risk-loving, risk-neutral or risk-loving. These propensities to risk represent an increasing acceptance of uncertainty in the QoS by the requester. The requester will participate in an SPR when Equation (3.4) is satisfied. For example, should $\overline{a} = 0$ then the requester requires that

[1]The non-functional property of availability is assumed to be defined over a continuous domain, and therefore represented by a continuous random variable

the expected utility from the SPR is positive.

$$EU \geq \overline{a} \tag{3.4}$$

This defines a calculative notion of trust for the requester in an SPR, where a requester has trust in a Web service to provide a QoS if the beliefs of the requester are such that the individual rationality constraint of the requester is satisfied. These beliefs are determined by the environment of an SPR, and under the assumption that the objectives and attitudes of the requester remain static, the participation of the requester in an SPR is influenced solely by these beliefs, which are derived from perception. The different environments for an SPR can reduce the uncertainty in the QoS of the Web service to different extents, and therefore increase trust to different extents. The requester can verify the fulfillment of the individual rationality constraint by different environments, and choose to participate in an SPR that environment which provides the greatest expectation of utility. For example, the ABC Investments may choose to participate in an SPR with XYZ Stockbrokers in the environment which reduces the uncertainty to the greatest extent, such that the individual rationality constraint is fulfilled and the expected utility of the requester is maximized. The institutions in the environment define the incentives for the provider to provide a certain QoS, but the requester must be able to verify the incentives which are sufficient for the provider to provide a certain QoS. This must implicitly considers the costs incurred by the provider, and the payment received. These obligations can define the appropriate incentives for the provider to provision the Web service with a certain QoS and for the participation of the requester.

3.6 Discussion

The institutional framework presented in this chapter provides a foundation on which methods for the formal verification of an environment can be performed. Such verification can prove the correctness of an environment with respect to the trust of a requester, and the participation of that requester in an SPR. The ability to perform such verification on the environment of an SPR would enable an environment to be verified for trust or absence of trust, under assumptions of the organizational objectives and beliefs of the requester, and therefore whether the requester would participate in an SPR within such an environment. This could inform the design of environments for SPRs to increase the trustworthiness of the environment, and the trust of the requester in this environment such that SPRs are prevented from failure due to the uncertainty in QoS experienced by the requester.

The formal verification of an environment requires methodologies to encode the holistic model of the environment described by the framework in this chapter. This encoding of the environment could then be utilized to verify whether the environment is trusted by the requester. The trust of the requester can be encoded as property of the environment such that the model of the environment

must exhibit this property. The property we are concerned with is the participation constraint of the requester, since this property dictates the participation of the requester in an SPR given his organizational objectives and beliefs. The environment is trusted when the individual rationality constraint is fulfilled by the environment under a given model of the requester. The model of the environment must consider the different layers in the environment and the different factors in the environment which can affect the fulfillment of the individual rationality constraint for the requester. Additionally, the model must consider the set of beliefs held by the requester at the point of the decision to participate in the SPR, as these beliefs define the perception of the environment and inform the decision to participate. The environment in which an SPR is formed dictates that a model of the environment and a model of the perception of this environment from the perspective of the requester is likely to have a high degree of complexity. There are numerous interacting factors in the environment which can together affect the trust of the requester. Accordingly, regardless of the methodology utilized for such formal verification, the computational complexity is likely to be high, and/or the veracity of the verification is likely to be low. The issue of computational complexity is a particular obstacle in the run-time verification of an environment by a requester, with such verification subject to temporal constraints. The design-time verification is more tolerant to high computational complexity, yet the problem of veracity remains.

Game theory [121, 181] is a mathematical framework for the modeling of agents which are interacting with one another and making interdependent decisions. Game theory provides a feasible methodology for the formal verification of the environment of an SPR. The SPR is modeled as a game between the requester and provider agents, such that their interaction leads to a certain set of outcomes. The interaction of these agents is assumed to be rational, such that the agents interact in accordance with self-interest and therefore to maximize expected utility. The beliefs, attitudes and objectives of the requester, the provider are explicitly represented by an equilibrium concept, with their execution and interaction dictated by the beliefs, attitudes and objectives encapsulated within their deliberative process. The payoffs for certain outcomes are determined by the communication and Platform Layers. For example, the presence of a monitor which can sanction a certain QoS reduces the payoff of the provider for the outcome where that QoS is provisioned.

The framework defined by game theory model the behavior of organization within an SPR from a global perspective. It "demonstrates the gains from cooperating and defecting in various contexts" [162] but is not able to provide "a theory of underlying costs of transacting and how these costs are altered by different institutional structures" [162]. Accordingly, there is no consideration in game theory of the "creation, evolution and consequence of rules" [182], focuses only on "modeling skills and strategies of players" [182]. Alternatively phrased, game theory models the reaction of organizations to a structure of rules rather than the formulation of the rules themselves. The rules are simply the determinant factors of the payoff structure which defines the game. The framework

can evaluate an existing environment but cannot define how to realize a better environment.

The rules define the utility for certain strategy profiles chosen by the requester and provider, but in order to consider the rules, the behavioral model of the organizations must incorporate these rules. That is, the requester and provider must be aware of the rules. This relies on a consistent perception of the environment of an SPR between the requester and provider, otherwise one may perceive rules which another does not, and essentially play a different game to the other. The behavioral model of the requesters and providers, under the rules defined in the institutions of the social context can be "contained in the description of the equilibrium" [171]. This behavior model can be represented by an appropriately defined equilibrium concept, which encapsulates all the rules to which the strategy selection of a social actor is subject. The framework enables the analysis of interaction from a global perspective under certain "rules of the game" but does not represent the rules only the consequences on the requester and provider, and common knowledge of these rules is assumed including the model of the requester or provider.

The ability to utilize an appropriate equilibrium concept to represent agents in an SPR requires the ability to formalize all the constraints to which the behavior of the agents are subject. These equilibrium concepts are at best a very generic approximation of behavior of computational agents, and only give an indication of how the game may be enacted. The formal verification is equivalent to the solving games with appropriate equilibrium concepts, where the equilibrium concepts may be computationally tractable. We need to be able to not only define the "rules of the game" but also how each agent reacts to the rules of the game, and moreover how each agent believes other agents will react to the rules of the game, *ad infinitum*. These rules can be defined in the form of what actions the requester and provider are obliged or permitted to perform in certain states of the game. Under the assumption that one can define an appropriate equilibrium concept, the game or service provisioning relationship in a given social context can be verified as to the properties which the predicted outcomes of the service provisioning relationship exhibit. The properties are defined as constraints over the equilibrium outcomes of the system, such that, in equilibrium, the system exhibits those properties. These properties are defined in terms of utility structures amongst agents. The equilibrium concepts have difficulty in representing in dynamic information, predicting the reaction of the agents to such dynamic information and the resultant effect on the outcome of the game. The verification is only a prediction under the abstract equilibrium concepts.

3.7 Conclusion

This chapter has addressed the problem of uncertainty in QoS experienced by the requester of a Web service. The main contribution of the chapter was a framework which facilitates the holistic representation of the environment of an SPR, and the subjective perception of this environment

by the requester. In the next chapter, this framework is utilized as the basis for the design and implementation of a software architecture for SPRs. This software architecture enables the practical consideration of the technical constraints to which organizations are bound, and demonstrates the efficacy and generality of the framework in the representation of environments.

Chapter 4

A Software Architecture For Service Provisioning Relationships

This chapter addresses the problem of uncertainty in QoS experienced by the requester and provider of a Web service. The chapter utilizes the framework presented in Chapter 3 to design and implement a software architecture which provides an environment for SPRs. The chapter has two main contributions. Firstly, a structured language for the representation of SLAs as contingent contracts, whose contingencies are defined over the QoS experienced by the requester. Such an approach differs from previous work, which does not provide a formal basis for the representation of SLAs as contingent contracts, where contingencies are defined in a verifiable and enforceable manner. The value of this approach is to enable the utilization of insights from economic literature on the design of contingent contracts to the creation of SLAs, such that the uncertainty of an organization in the QoS can be reduced. Secondly, a practical methodology for the creation of SLAs as contingent contracts, whose contingencies are optimal with regard to the objectives of an organization, in the presence of uncertainty in QoS. Such an approach differs from previous work, which does not provide a generic process for the creation of SLAs for any QoS metric of any Web service, given organizational objectives. The value of such an approach is to provide a formal process for the creation of optimal SLAs, such that organizations no longer utilize processes based on ad-hoc, sub-optimal reasoning which can be inconsistent with organizational objectives.

The remainder of this chapter is structured as follows. Section 4.1 introduces the problem addressed in this chapter. Section 4.2 presents a set of functional and non-functional requirements for the software architecture to address this problem. Section 4.3 presents the assumptions of the architecture. Section 4.4 describes the design of the software architecture to fulfill the set of requirements. Section 4.5 presents the implementation of the architecture. Section 4.6 presents two different deployments of the software architecture. Section 4.7 evaluates the software architecture. Finally, the chapter is briefly summarized and concluded in Section 4.8.

4.1 Introduction

Uncertainty in QoS can be experienced by the requester and provider of a Web service. The requester of a Web service can be uncertain of the willingness and ability of a provider to provision a Web service with a certain QoS. This uncertainty arises from a lack of complete control over, and complete information pertaining to, the Web service. Additionally, the provider of a Web service can be uncertain of his ability to provision a Web service with a certain QoS. This uncertainty arises from a lack of complete control over, and complete information pertaining to, the computing infrastructure on which the Web service is provisioned. Such uncertainties leaves the requester and provider vulnerable to the provision of a Web service with a QoS which does not meet the requirements of a requester, leading to negative effects on the fulfillment of organizational objectives. These uncertainties can endanger the economic viability of SPRs, and mitigate any increases in organizational efficiency for the requester and provider. The organizations may prefer not to participate in SPRs given the presence of such uncertainties, due to the negative effects on the fulfillment of organizational objectives from usage or provision of the Web service with a certain QoS. Accordingly, the economic viability of SPRs is predicated on the ability of organizations to address such uncertainties.

SLAs are often formed between the requester and provider in an SPR, in order to formalize the obligations of each organization and reduce uncertainty in QoS. These SLAs can be formulated as regulative institutions between the requester and provider, with formal sanctions for breaches of contractual terms to incentivize a certain QoS. The contractual terms within such regulative institutions can take many different forms dependent on the extent of the uncertainty which is to be overcome by the organizations. Contingent contracts [183] are a specific class of SLA which can address the uncertainty in QoS experienced by both organizations. Such contracts comprise of a set of contingencies, representing the set of outcomes for an uncertain event, and contractual terms on the occurrence of these contingencies. The uncertain event in an SPR is the provision of a Web service with a certain QoS, and the set of contingencies are the different feasible levels of QoS with which the Web service may be provisioned, along with the contractual terms for each level of QoS, such as monetary payments. The outcomes which represent each contingency must be verifiable, such that the actual outcome can be unambiguously determined, and the appropriate contingency enacted within the contract. Such contracts can provide motivation for the provider to provision the Web service with a certain QoS, can facilitate the shifting of uncertainty in QoS between the requester and provider, and can overcome the informational advantage of the provider with regard to the expected QoS. In essence, such contracts make the differences in expectation of the QoS "the basis of a bet" [183] between the requester and provider, and are particularly useful in domains where highly dynamic relationships are formed between organizations with no previous relationships.

In accordance with the framework defined in Chapter 3, contingent contracts must be defined and enforced by Web services at the Platform Layer. This Platform Layer can then be deployed on a certain Communication Layer to provide constraints for any Web service defined at the Service Layer, such as the Stock Quote Service. The Web services required to realize an environment for such institutions is an aspect of contingent contracts which has not been addressed in previous work such [134, 159]. Such work focuses on the process by which contractual terms should be instantiated to shift uncertainty between organizations and overcome informational advantages. There are a number of interdependent Web services required to realize the environment to support an SLA as a contingent contract. These Web services must facilitate the creation and negotiation of such SLAs where the contingencies and contractual terms reflect the objectives of an organization, such as risk-sharing or mitigation of information advantage. Such Web services must facilitate the monitoring of the QoS to facilitate enactment of the appropriate contingencies in these SLAs, and must define the contingencies in a manner which is verifiable using commonly understood, unambiguous semantics. Additionally, there must exist Web services which provide resources that can be utilized for the purposes of sanctioning, such as monetary payments. Accordingly, this chapter presents a software architecture which realizes the Web services required in an environment for SPRs based upon contingent contracts. Such instantiation of the framework from Chapter 3 provides a demonstration of the expressiveness and generality of the framework, and enables the technical considerations of realizing such Web services, and the communication between these Web services to be investigated.

4.2 Requirements

The requirements of a software architecture refer to the "information, processing and the characteristics of that information and processing required by the user of the system" [184], and include both functional and non-functional aspects of the software architecture. Functional requirements refer to the desired behavior of an architecture, defining the set of functions which encapsulate the information and processing required of the system. Non-functional requirements refer to the qualitative properties [185, 186, 187] with which the efficacy of the functionality provided by the system can be assessed. The requirements of the software architecture which is defined in this chapter are as follows:

4.2.1 Functional Requirements

The functional requirements of the software architecture are defined as follows:

1. **A structured language for the representation of SLAs**

   The software architecture must provide a structured language for the explicit representation of

SLAs formulated as contingent contracts, where QoS is defined with precise well-understood semantics and is expressed in a verifiable and enforceable manner.

2. **A Web service for the creation and negotiation of SLAs**
   The software architecture must provide a Web service for the creation of SLAs formulated as contingent contracts whose contingencies are optimal in relation to certain organizational objectives, and enable negotiation of the SLA such that the contingencies remain consistent with organizational objectives.

3. **A Web service for the monitoring of QoS**
   The software architecture must provide a Web service for the monitoring of QoS within an SPR, to facilitate the enforcement of an SLA and the enactment of contingencies based on the QoS.

4.2.2 Non-Functional Requirements

The functional requirements of the software architecture are defined as follows:

1. **Simplicity of SLAs**
   The representation of SLAs must be sufficiently simple to facilitate straightforward design and evaluation of contingencies with regard to organizational objectives.

2. **Adaptability to changes in objectives.**
   The Web services within the software architecture must be easily adaptable to changes in organizational objectives, such as changes in management costs or monetary payments.

3. **Extensibility to different Web services and different metrics**
   The Web services within the software architecture must be easily extensible to the creation and monitoring of SLAs for different Web services and different QoS, and must not require any changes to these Web services.

4.3 Architectural Assumptions

The framework presented in this chapter is based upon a set of assumptions which include those defined by the framework in Chapter 3 and the following additional assumptions:

1. **The SLA contains one SLO for a single usage of a Web service**
   The SLAs within the software architecture contain one SLO defined for a single metric, and for a single immediate usage of a Web service. These SLOs define a binary set of outcomes for a metric, representing the fulfillment and violation of a certain value for the metric.

2. **A Web service is utilized immediately after negotiation of an SLA**
   The usage of the Web service immediately follows the negotiation of an SLA, in a manner consistent with spot pricing [188]. Such consumption considers Web services to be commodities, with identical and substitutable Web services provisioned by different providers.

3. **The impact of the communication network and the Web service on the QoS can be distinguished**
   The QoS experienced by the requester is a function of the QoS of the Web service and the QoS of the communication network. This approach assumes an ability to distinguish the impact of the communication network from the impact of the Web service on the QoS experienced in some (approximate) manner.

4.4 Design

The design of a software architecture describes the "modularization and detailed interfaces of the design elements, their algorithms and procedures, and the data types" [184], such that the functional and non-functional requirements of the architecture are fulfilled. In this section, we provide a summary of the architectural style which will be utilized for the software architecture, present a set of architectural prerequisites, and describe the Web services which comprise the architecture in accordance with the chosen architectural style.

4.4.1 Architectural Style

Web services can adhere to different architectural styles, which define a "pattern for construction" [189]. These styles impose constraints on interaction with Web services to induce desirable non-functional properties. There are two common architectural styles utilized for Web services: Representational State Transfer (REST) [190] and Service Oriented Architecture (SOA) [191, 192]. These architectural styles impose different constraints on the interaction with a Web service. The functional requirements of the software architecture (see Section 4.2.1) can be fulfilled by both architectural styles, with appropriate design of the architectural components. The key distinction between these architectural styles arises from the extent to which certain non-functional properties are exhibited by these architectural components, and the resultant software architecture. The architectural style of REST has been chosen for the software architecture. Such a selection is of particular interest given the absence of any software architecture in previous work which utilizes the REST style. Therefore, the software architecture defined in this chapter can be considered a proof-of-concept for the utilization of this architectural style for such a software architecture.

REST necessitates the definition of a uniform interface to define interaction with resources.

We consider a generic, uniform interface based on four state-centric operations. These operations facilitate introspection on, and manipulation of the state of a resource through some representation of that state: `create()`, `retrieve`, `update()`, and `delete()`. A more concise set of operations consisting of `retrieve()` and `update()` could be utilized for resources, but such a set of operations makes certain semantics involved with the manipulation of state implicit, namely `create()` and `delete()`. This places more relevance on the representation to determine the semantics of a response. The operations can be mapped to any concrete communication protocol in a specific implementation of the software architecture. The functionality provisioned by any Web service in the software architecture must conform to this uniform interface, and provide the set of state-centric operations for interaction with the functionality through this resource. The requests to the Web service contain the operation to perform on the resource along with auxiliary data, including the representation to utilize in the request (if applicable) and any metadata pertaining this representation. The responses from the Web service can also contain auxiliary data, including the representation resulting from the request (if applicable) and any metadata pertaining to the resource or the representation of the resource. The design of the architecture is presented with a emphasis on the data flow of requests and responses transmitted along pipes through a "series of computational or manipulative processing components" [189], consistent with a pipe-and-filter pattern.

4.4.2 Architectural Prerequisites

The architectural prerequisites define two Web services on which the Web services defined in Section 4.4.3 are dependent. These pre-requisite Web services provide general functionality required by the software architecture, but whose design and implementation is omitted in this thesis. Such omission is motivated by the existence of varied commercial Web services provisioning such functionality which could be utilized within the software architecture. This avoids the design and implementation of Web services to provide this functionality which simply re-implement existing functionality. It is assumed that such services pre-exist in the environment, and are provisioned by a TTP, such that the functionality provisioned by these services is not subject to any uncertainty. In the next two sections, we describe these two Web services, and provide an overview of their utilization within the software architecture.

4.4.2.1 Identity Service

The Identity Service provides the functionality to authorize and authentication organizations within a domain. Authorization enables the Web service to determine the organization performing an interaction with a Web service. For example, an interaction with a Web service can be associated with ABC Investments. Authentication enables the Web services to determine the properties of

the organization performing an interaction. For example, ABC Investments has the appropriate properties to interact with the the Stock Quote Service of XYZ Stockbrokers.

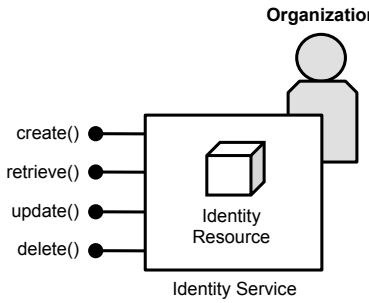

Figure 4.1: An Illustration Of The Identity Service

Figure 4.1 illustrates an Identity Service, defined in accordance with REST style. Such adherence to the REST style is not mandatory but provides consistency for the Web services defined within the software architecture. A Web service outsources the functionality to authorize and authenticate organizations to the Identity Service, enabling the Web service to specialize in the provision of functionality predicated on this notion of identity, such as the creation of an SLA. The Identity Resource provides a representation of the identity of a real-world organization. This representation includes properties of that real-world organization, as verified by the Identity Service, such as name and address. This representation can be retrieved by other Web services for authentication, or may be manipulated by other Web services to change the properties of the organization in accordance with updated information. The functionality of authorization and authentication is a pre-requisite of the software architecture due to the requirement to provide accountability for the interactions of organizations within an SPR, and to define obligations on organizations given this accountability. In the absence of a persistent notion of identity for the organizations, there is no basis on which different organizations can be distinguished or held accountable for their interactions retrospectively. Commercial examples of Identity Services include those provided by Verisign [91], Google Accounts [109] and Facebook [193]. Such Identity Services can be based upon proprietary standards for the representation of identity, and methods of authentication and authorization, or or open standards such as Open ID [194]. Conventionally, these Web services link to a real-world notion of identity, such that a certain identity within the Web service corresponds to a real-world organization or individual. This facilitates the utilization of real-world institutions to constrain the behavior of organizations in an SPR in a similar manner to constraints applied in real-world outsourcing relationships.

4.4.2.2 Payment Service

The Payment Service provides the functionality to transfer monetary payment between organizations in a domain. These monetary payments between organization provides a medium of exchange,

and common store of value between organizations. For example, ABC Investments can transfer payment to XYZ Stockbrokers for usage of the Stock Quote Service. The functionality of the transfer of monetary payments enables value to be transferred between organizations and eliminates the problems of barter exchange, and enables monetary sanctions to be imposed on organizations for certain interactions with a Web service. For example, XYZ Stockbrokers may transfer a payment to ABC Investments as part of a sanction for a certain QoS.

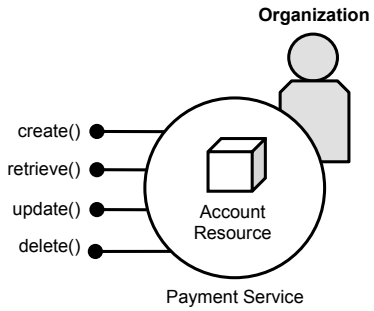

Figure 4.2: An Illustration Of The Payment Service

Figure 4.2 illustrates a Payment Service, defined in accordance with the REST style, which, in an analogous manner to the Identity Service, provides consistency amongst the Web services in the software architecture. A Web service outsources the functionality to transfer monetary payments between organizations to the Payment Service, enabling the Web service to specialize in the provision of functionality predicated on this notion of payment, such as the monitoring and enforcement of SLAs. Commercial examples of Payment Services include those provided by Paypal [195] and Western Union [196]. Conventionally, these Web services link to a real-world notion of money, such that the monetary transfers performed within the Web service correspond to actual monetary transfers between real-world organizations or individuals. Indeed, the Payment Service is predicated on the existence of the Identity Service, in order to be able to authorize and authenticate monetary transfers between organizations. The Account Resource represents the set of monetary payments for an organization, and can have a representation which includes some or all of these monetary payments and a balance of payments. This representation can be retrieved by the organization to which it pertains, or may be manipulated by other Web services to change the balance. Such retrieval and manipulation requires appropriate authorization and authentication for organizations, such that the manner in which the account resource can be manipulated is based upon the organization performing the manipulation. For example, any organization may be able to transfer payments to an account resource, whilst only a specific set of trusted organizations, such as the Government, can transfer payments from an Account Resource.

4.4.3 Architectural Components

The architectural components whose design is presented in this section, in addition to the architectural prerequisites defined in Section 4.4.3, constitute the platform layer on which an SPR for a specific Web service can be supported.

4.4.3.1 Service Level Agreement

The Service Level Agreement component is a data component which defines the language and structure for the explicit representation of SLAs which expresses QoS in a verifiable manner, and whose terms refer to precise well-understood semantics for organizations. This language and structure was defined with the verifiability of outcomes and the enforceability of SLAs as contingent contracts. Previous work on SLAs discussed in Chapter 2 often focuses on SLAs from a management rather than a trust perspective, often utilizing SLOs which focus on system metrics of the provider rather than experience metrics of the provider.

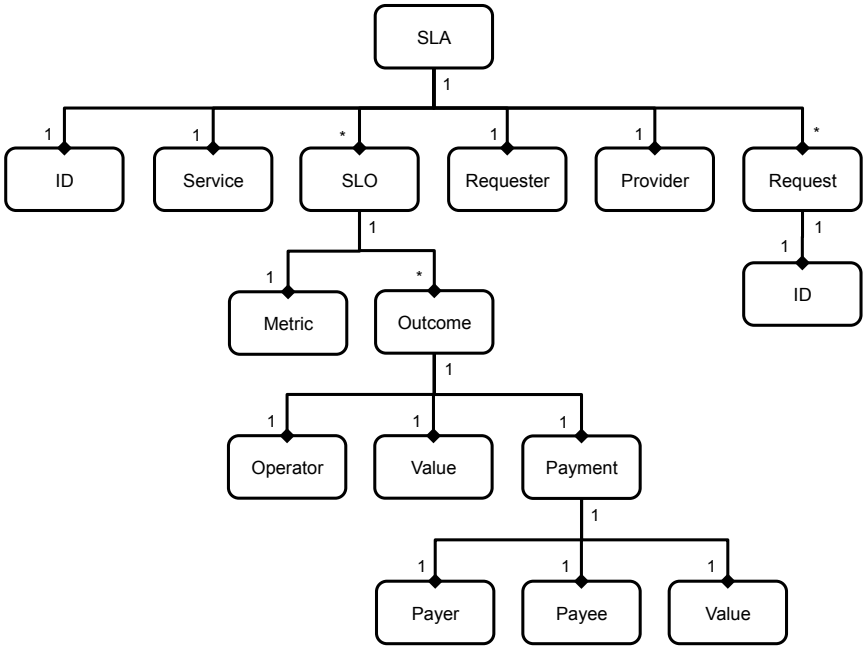

Figure 4.3: An Entity Relationship Diagram For A SLA

Figure 4.3 provides an entity relationship diagram to describe the language and structure chosen for an SLA. The SLA contains an identifier to facilitate the unique identification of a particular SLA. The organizations within the SPR defined by the SLA are uniquely identified within the SLA, along with the Web service to which the SLA pertains. This unique identification can utilized an Identity Service as described in Section 4.4.2.1. The QoS is within a set of SLOs, where an SLO contains the identifier of the metric to which the obligations pertain, a set of outcomes over this metric

(defined by a binary operator and a value), and a monetary payment between the organizations for the realization of this specific outcome for each service usage request. This payment can utilize a Payment Service as described in Section 4.4.2.2. The outcomes for a metric partitioned the domain of values for a metric into disjoint sets, such that there is no ambiguity with regard to the outcome. This domain of values can be defined over a discrete domain such that the binary operator utilized for a outcome is $=$, or can be defined over a continuous domain such that the binary operator utilized for an outcome is $<$, $\leq$, $\geq$, or $>$.

There are a set of requests defined within the SLA, which uniquely identify the requests to which the SLA pertains, such that the outcomes defined in the SLA apply to each of these requests. This is distinct from previous work on SLAs which often define the validity period of an SLA with a time period. Such utilization of time is troublesome as requests within a period have no unique reference for the organizations in the SPR, such that QoS cannot be verified for each individual request.

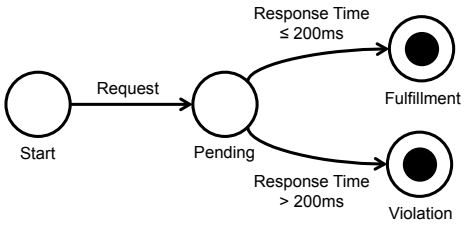

Figure 4.4: A Finite State Automata For Response Time

Figure 4.4 illustrates the ability to model each request utilizing a Finite State Automata (FSA) [197]. The transitions of the FSA represent the interaction with the Web service through messages, with each transition representing one message. The message can be request to the Web service, or a response from the Web service. The metrics are defined over the final states of the FSA, which in this case are the outcomes representing a response time of less than or equal to 200 milliseconds, or greater than 200 milliseconds. For any request, the state machine has a final state which represents the outcome in terms of QoS.

The ability to verify an outcome is dependent on the ability to detect the request messages to, and response messages from the Web service, and determine the final state based upon these messages. We choose to distinguish two classes of metric for a single request to a Web service. The composition of these two classes of metric define the space of metrics which can be defined for each request. A boolean metric is a metric where the final state refers to the semantics of the response. For example, a boolean metric to represent availability would have two final states: available and unavailable which can be determined from the semantics of the response from the Web service. A temporal metric is a metric where the final state refers to time taken to transition to a final state. For example, a temporal metric to represent response time would have n final states, where n represents the number of intervals into which the domain of values for response time has been partitioned to

distinguish the outcomes. Such metrics are derived through the performance of time-stamping at each state in the FSA. The semantics of messages utilized to define the domain of values for a boolean metric can be based on those semantics which can be unambiguously defined by the organizations in the SPR. These semantics can be mapped to the semantics of the communication protocol on which messages are transmitted, or the semantics of the messages themselves. For example, a Web service based on SOA which utilizes SOAP could define metrics based on the semantics of the SOAP messages, or the semantics of the underlying communication protocol on which the SOAP messages are communicated.

4.4.3.2 Agreement Service

The Agreement Service provides the functionality for the creation and negotiation of SLAs whose contingencies are optimal in relation to the beliefs and objectives of the organization provisioning the Web service. For example, XYZ Stockbrokers can create a SLA for the Stock Quote Service which maximizes the expected positive payment from participation in the SPR with ABC Investments. The Web service defined in this section utilizes the language and structure of SLAs defined in Section 4.4.3.1.

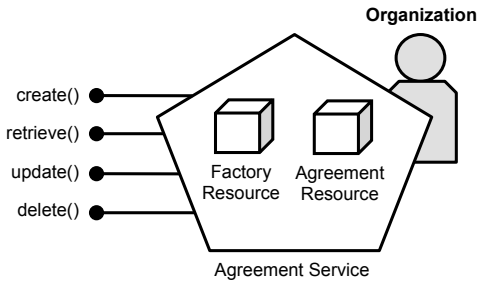

Figure 4.5: An Illustration Of The Agreement Service

Figure 4.5 illustrates the two resources which comprise the Agreement Service: Factory Resource and Agreement Resource. The Factory Resource encapsulates the functionality for the creation of a SLA on request. For example, a requester could request the creation of a SLA from the provider of a Web service. This SLA can change in accordance with the beliefs and objectives of the organization such that a request for the creation of a SLA at two different points in time yields SLAs with different SLOs. For example, the beliefs of the organization over a certain metric may change, leading to a change in the SLOs which are created for that metric which is reflected in the state of the factory resource.

The Agreement Resource encapsulates the functionality for the negotiation of a SLA. The request to the Factory Resource creates a unique resource to represent that specific SLA. The requests to the agreement resource facilitate introspection on the SLA, and manipulation of the SLA as part

of a negotiation process. For example, a request to the Agreement Resource can update the state of the SLA represented that resource to contain different SLOs. The acceptance of such updates is defined by the beliefs and objectives of the organization which provisions the Agreement Service. For example, a provider will only accept an update to the state of the SLA if such an update is consistent with his objectives. In order to facilitate the creation of SLAs, and negotiation of SLA in accordance with beliefs and objectives. The beliefs, attitudes and objectives of the organization provisioning the Agreement Service must be defined in an appropriate manner. This information is contained with in the configuration supplied to the Web service prior to its instantiation.

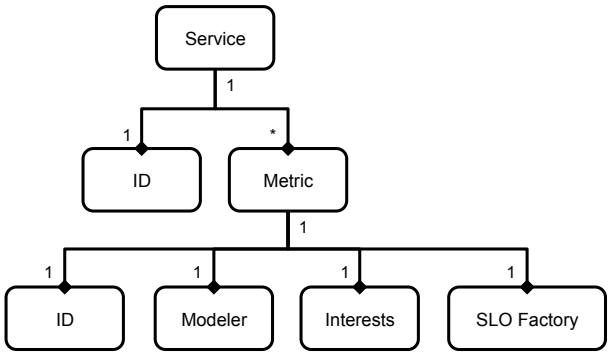

Figure 4.6: An Illustration Of The Agreement Service Configuration

Figure 4.6 defines the information which is utilized to configure the Agreement Service. The configuration contains the set of Web services for which the creation of SLA is supported. These Web services are uniquely identified such that their functional semantics are unambiguously defined for organizations. For each Web service, the Agreement Service defines a set of metric for which the creation of SLOs in a SLA for the Web service are supported. These metrics are uniquely identified such that their semantics are unambiguously defined for organizations. We discuss such unambiguous definition of metrics in Section 4.4.3.3. A modeler is associated with each metric, which defines the manner in which the beliefs of the organization with regard to this metric are obtained. This modeler can utilize any theoretical or empirical techniques to model the values of the metric and produce a probability distribution over values of the metric. This probability distribution represents the beliefs of the organization.

The objectives of the organization over the values of the metric are defined within two different functions: fulfillment and violation. These functions define monetary payments for the organization for a fulfillment outcome and violation outcome respectively, under the assumption that a SLO contains two contingencies for any metric representing fulfillment and violation of a QoS. The fulfillment and violation functions are static, such that the monetary payments are not contingent on the actual value of the metric experienced, only that the value of the metric was within the domain of values which constitutes fulfillment or violation. This yields a step function representing the objectives of

the organization across the domain of possible values for a metric. Finally, the SLO factory defines the disposition of the organization with regard to uncertainty, defining the manner in which the beliefs and objectives are combined into the optimization problem which can be subsequently solved for the metric to yield optimal SLOs.

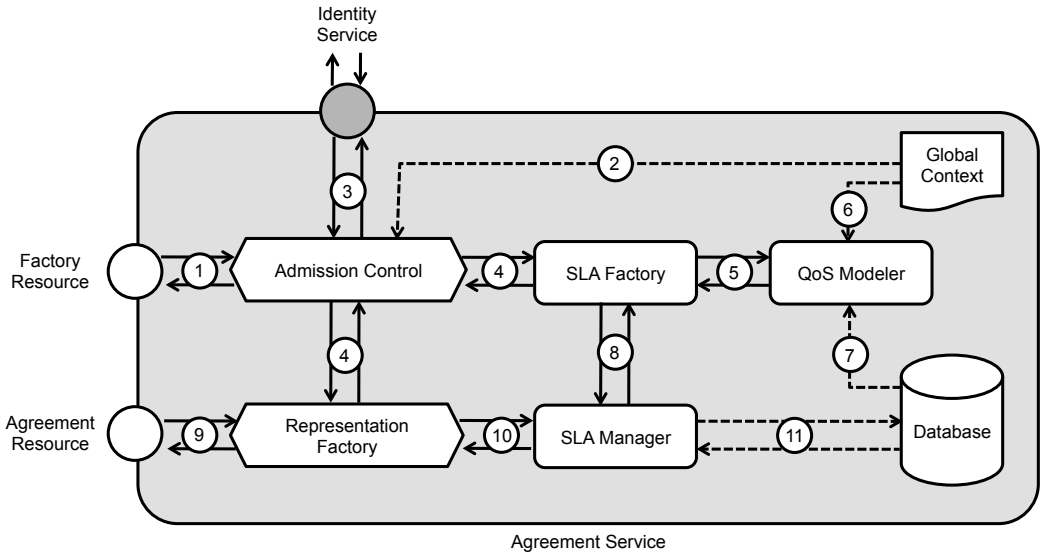

Figure 4.7: A Data Flow Diagram For The Agreement Service

Figure 4.7 provides a data flow diagram which illustrates the design of the Agreement Service, consisting of the Factory and Agreement Resource. The diagram describes the flow of requests and responses between internal components of the Web service in order to realize the creation and negotiation of SLAs. The Admission Control component is a filter of requests which performs authorization, using the Identity Service, such the SLAs can only be created for, retrieved and manipulated by a certain organization. This component can control the admission of requests for SLAs based on information such as the current load on the Web service. This information is stored in the Global Context of the Web service. The representation factory component is a filter of requests and responses which performs transformation between the representation of a SLA communicated over the network, and the representation of an SLA as a run-time object within the Agreement Service.

The QoS modeler is a processing component which derives the beliefs of the organization for utilization within the creation of an SLA. The modeler utilizes the method defined for each metric in the configuration of the Agreement Service to derive a probability distribution utilized by the SLA Factory in the creation of SLOs for that metric. These modelers can utilize information within the Global Context to inform the modeling process. This is particularly pertinent when the Agreement Service can access information on the current state of the Web service, obtained from the Monitor Service (discussed in Section 4.4.3.3).

The SLA Factory is the principal processing component in the Agreement Service, and performs the optimization process to derive the optimal SLA. The SLA Factory contains the objectives of the organization defined for every metric, or every Web service for which the creation of SLAs is supported. These objectives are combined with the beliefs obtained from the QoS modeler to form an expected utility function. This expected utility function can be utilized to calculate the expected utility of any SLO. The SLO Factory for each metric can utilize this ability to calculate expected utility over any SLA in order to derive the optimal SLO, over a set of potential SLOs. The set of potential SLOs is the domain over which a metric is defined. For example, a response time metric, m, would be defined over the domain of real numbers, $v \in \mathbb{R}$. The domain over which a metric is defined can be partitioned up into any number of outcomes in accordance with the granularity of SLOs required. Each of these outcomes can have different associated utilities such that the optimization process must optimize across the beliefs in which outcomes will occur and the utilities for these outcomes.

input : Metric, m
input : Desires, $g_m(v)$ and $h_m(v)$
input : Beliefs, $\overline{f_m}(v)$
output: Optimal SLA, a^*

$eu \leftarrow$ `EUBuild(`$g_m(v), h_m(v), \overline{f_m}(v)$`)`;
$dv \leftarrow$ `Differentiate(`eu, v`)`;
$v^* \leftarrow$ `BisectionMethod(`dv`)`;
$a^* \leftarrow ((m, v^*), g_m(v^*), h_m(v^*))$;

Figure 4.8: An Algorithm For Optimal SLA Creation

Figure 4.8 illustrates the algorithm utilized to perform the optimization over the domain of SLOs. The optimization of SLOs for each metric yields an optimal SLA where the expected utility of the organization is maximized. The objectives are represented by the two functions: $g_m(v)$ and $h_m(v)$ which represent fulfillment and violation function respectively. These functions are assumed to be linear polynomials to ease the differentiation in order to find the first-order derivative of the function and perform optimization. In order to find the optimal value of the metric, we find the turning or stationary points of the first-order derivative where the derivative is equal to zero. The bisection method of root-finding was utilized, which takes as input the function whose root is to be found, the lower and upper bounds of this function, the maximum number of iterations to be performed in root-finding, and the acceptable error in the root-finding. The maximum number of iterations and the acceptable error determine the computational tractability of the algorithm, with a greater number of iterations and lower higher leading to high consumption of time and computational resources by the algorithm. The output is the value of the differentiated expected utility function which evaluates

to zero, or the root of the function, which represents the optimal value for the metric.

The creation of a SLA instantiates an Agreement Resource to represent to SLA and facilitate introspection or manipulation. The SLA manager component enables this introspection and manipulation through the provision of the functionality to store SLA in the persistent storage of the database, retrieval SLAs from the persistent storage, create a run-time object for the SLA, perform any processing on the run-time object to reflect negotiation, and then returning the updated representation of the SLA in the response. The SLA manager must be able to check that the any manipulation of the SLA within the negotiation process is valid. The SLA manager can interact with the SLA factory in order to check the expected utility of a proposed manipulation of the SLA, and verify that a minimum level of expected utility is yielded. Clearly, such negotiation will yield a SLA which is no longer optimal, but may still be preferable for the organization than the absence of participation in a SPR. The process of negotiation is assumed to be completed, and the SLA committed to, when the proposed SLA is equal to the current state of the SLA.

4.4.3.3 Monitor Service

The Monitor Service provides the functionality the monitoring of QoS within the SPR defined by the SLA to facilitate the enactment of contingencies in the SLA based on the QoS. This service is an intermediary on the communication network between the requester and provider in the SPR, and can monitor the QoS for interaction without influence on the semantics of those requests and responses. For example, CDE Trust monitor the QoS in the SPR between ABC Investments and XYZ Stockbrokers for the Stock Quote Service, and enactment the contingencies in the SLA based on the QoS.

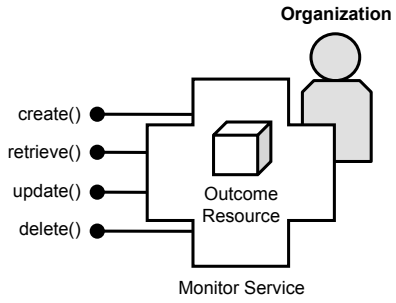

Figure 4.9: An Illustration Of The Monitor Service

Figure 4.9 illustrates the Monitor Service. The Monitor Service exposes a set of outcome resources which encapsulates the state of a request defined within a SLA. The requests to the outcome resource represent requests to the underlying Web service for which the monitoring is performed. The context of a request, such as the Web service to which the request pertains and the SLOs to which the request is subject can be derived from interaction with the Agreement Service.

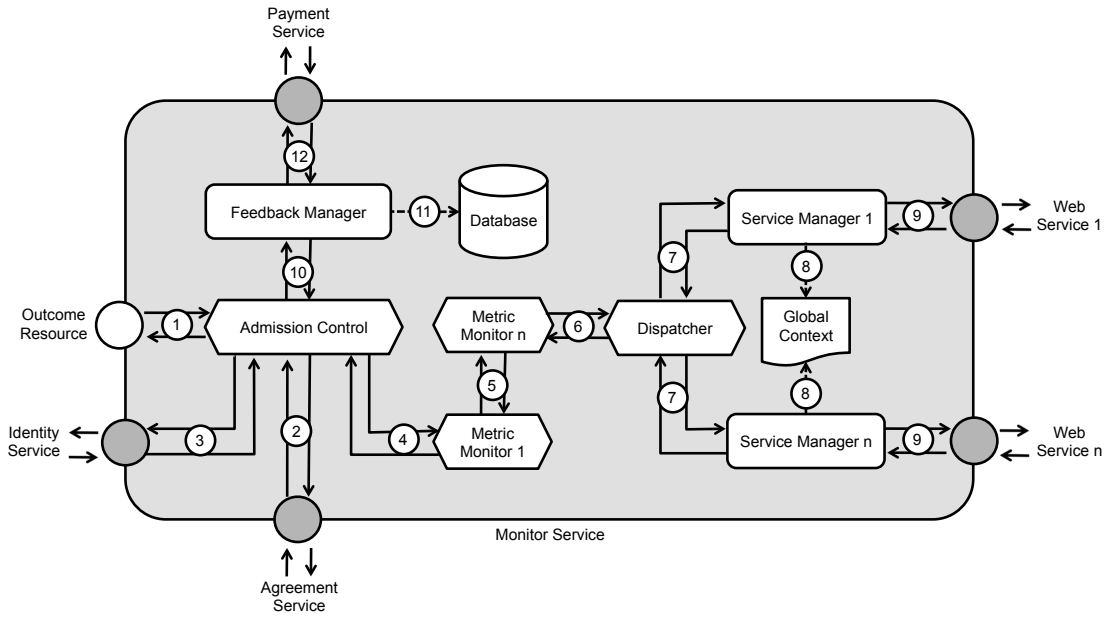

Figure 4.10: A Data Flow Diagram For The Monitor Service

Figure 4.7 provides a data flow diagram illustrating the design of the Monitor Service in order to provide the functionality to monitor the QoS of a Web service. The data flow diagram illustrates the flow of requests and responses between the components in the Web service and the Web services discussed in the Section 4.4.2. The Admission Control performs authorization and authentication of requests utilizing the Identity Service. This enables the Monitor Service to verify the identity of the organization, and to utilize the Agreement Service to retrieve the SLA to which the request pertains, and verify that the organization is the requester in the SLA. Additionally, the Admission Control component verifies that a request with the same identifier has not been previously sent to the Monitor Service. This SLA information is associated with the context of the request in the Monitor Service, such that it can be utilized subsequent to the processing of the request to enactment appropriate contingencies.

The request for the Web service is passes through a set of Metric Monitor components, where each Metric Monitor represents the monitor of a specific metric. This enables metrics to be easily added and removed as necessary, and provides a clear distinction amongst metrics and their semantics. The Metric Monitor for each metric is simply defined using the state machine representation defined Section 4.4.3.1. The monitor detects a request and performs any pre-processing on the request (storing this in the context of request), forwards the request to the Web service, and then performs any post-processing on the response which involves the detection of the final state of the request. This may involve introspection on the semantics of the response (boolean metric) or may involve another time-stamp to calculate a time for the request-response pair (temporal metric).

The Dispatcher holds the set of services for which SLAs can be monitored. The context of the

request is inspected to derive the Web service to which the request must be dispatched. The Dispatcher is initialized with all the services for which monitoring is supported, and associated with each Web service a Service Manager which processes the requests for that specific Web service. The Service Manager performs any pre-processing on the request before dispatching the request to the service for processing. The Service Manager then performs any post-processing on the response before forwarding the response onto the Metric Monitors for verification of the QoS. The Service Manager separates the requests for the Web service from the resources on which these requests are processed. This enables the requests to be unconcerned about where they are processed, and enables workload management such as load balancing to take place, or the utilization of different providers for functionally analogous Web services for the purposes of fault-tolerance. The functionality of the Service Manager is dependent on the organization provisioning the Monitor Service. The Service Managers can update the information in the Global Context for utilization by, for example, Admission Control in the Agreement Service (when both services are co-located at the same organization). We describe the design of two specific Service Managers in the software architecture: Grid Service Manager and Cloud Service Manager.

- **Grid Service Manager**

  The Grid Service Manager enables the monitoring of QoS for a Web service provisioned on a Grid computing infrastructure. The Grid Service Manager enables a Web service to be deployed on a Grid computing infrastructure such that the same Web service is replicated over a physical resource capacity. Figure 4.11 illustrates the Grid Service Manager. The Grid Service Manager contains a pool of resources on which a certain Web service is deployed, such that requests for that service can be dispatched to those resources. The requests to the service are held in a FIFO queue before being dispatched to a resource when available.

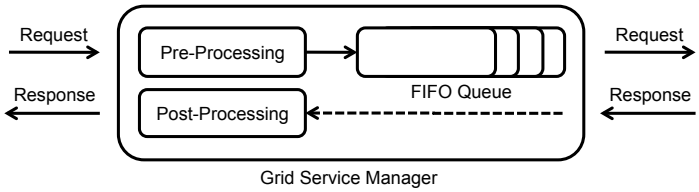

Figure 4.11: An Illustration Of The Grid Service Manager

- **Cloud Service Manager**

  The Cloud Service Manager enables the monitoring of QoS for a Web service provisioned on a Cloud computing infrastructure. The Cloud Service Manager enables a Web service to be deployed on a Cloud computing infrastructure such that the Web service is replicated over a logical resource capacity. Figure 4.12 illustrates the Cloud Service Manager. The Cloud Service Manager contains any necessary pre-processing required to dispatch requests to the

cloud service. This can include the transformation of the request body into a specific format required by the API of the cloud service. Such transformation may be required in the response received from the service.

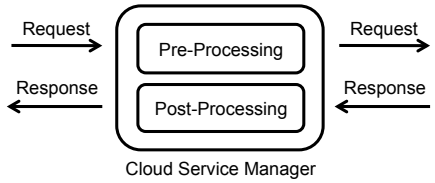

Figure 4.12: An Illustration Of The Cloud Service Manager

The Feedback Manager deals with the enactment of contingencies in the SLA for a certain outcome. The Metric Monitors store the outcome for each metric within the context of the request and this request is passed to the Feedback Manager which can enact contingencies in the SLA based upon these outcomes. The outcomes are also written to persistent storage, for instance, to inform future creation of SLAs in the Agreement Service (when both services are co-located at the same organization), or to update beliefs as part of a trust mechanism based upon learning. Indeed, the Feedback Manager can provide a representation of the outcome resource subsequent to the processing of the request to which it pertains, which represents the QoS for that request. We do not consider such a representation in this work.

4.5 Implementation

The implementation of a software architecture provides the "representations of the algorithms and data types" [184] described in the design, such that the functional requirements and non-functional requirements of the architecture are fulfilled. In this section, we describe the implementation of the architectural components, the deployment of these architectural components for Web services on two different computational infrastructures, and the software tools utilized to implement the software architecture.

4.5.1 Architectural Components

The implementation of the architectural components in Section 4.4.3 is presented in this section.

4.5.1.1 Service Level Agreement

The Service Level Agreement component designed in Section 4.4.3.1 is implemented as a run-time object utilizing a hierarchy of dependent Java classes. These Java classes encapsulate the data

which represents the SLA, and a set of operations for introspection on, and manipulation of this data, which can be utilized by the Web services within the software architecture.

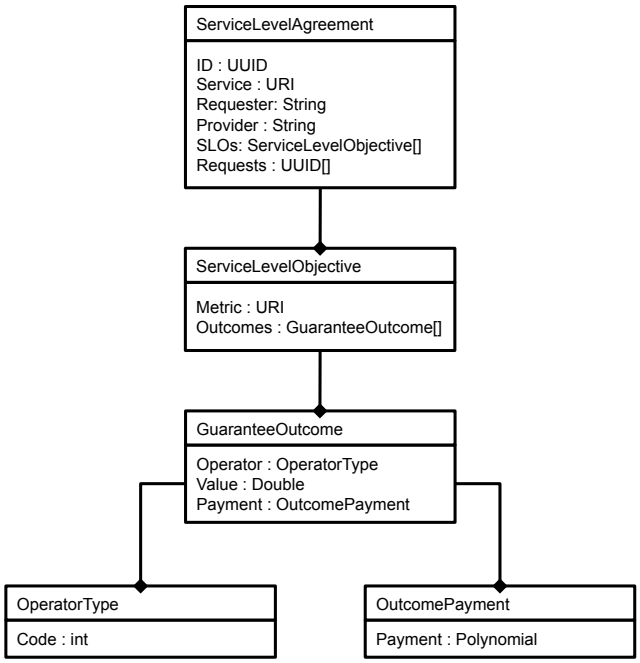

Figure 4.13: A Class Diagram For An SLA

Figure 4.13 illustrates the primary components in the hierarchy of classes which implement the SLA data component. This class diagram provides a concrete implementation of the entity relationship diagram illustrated in Figure 4.3, with the selection of appropriate data types. The `ServiceLevelAgreement` class represents the top of the hierarchy of classes, encapsulating all the data which represents the SLA. The ID of the SLA is represented by a Universal Unique Identifier (UUID) [198] which is created at random by the Agreement Service on instantiation of an SLA object. This data type enables an ID to be created for each SLA without fear of collisions amongst the IDs created for these SLAs, and therefore without fear of ambiguity. The service, organizations in the SLA are represented by a URI. This representation enables the service, organizations to be identified by a resource. The SLOs in the SLA are contained within an array of `ServiceLevelObjective`.

The `ServiceLevelObjective` class contains the metric to which the SLO pertains, where the metric is represented by a URI for analogous reasons to the service, organizations. An array of `GuaranteeOutcome` is defined which contains an operator over the metric, a value for the metric in the form of a double such as 0.10, and a payment. The operator is encapsulated within a `OperatorType` class, which enables the construction of a set of finite set of binary operators such as $=$, $\leq$, or $\geq$. The payment is represented by an `OutcomePayment` class which defines the payment as a polynomial function, whose single variable was the value of the metric over which the payment

was defined.

The communication of the SLA between organizations on a communication network necessitates an appropriate encoding of the SLA in a data format suitable for transmission over that network. Accordingly, a XML representation is defined for SLAs based upon XML. XML provides a simple and general set of rules for encoding documents as hierarchically structured text for transmission over a communication network. An alternative language for the representation of the SLA is Javascript Object Notation (JSON) [199], which defines a lightweight language in for encoding documents as groups of name-value pairs. The generality of XML and the software support for its utilization motivate its selection within the software architecture. The utilization of XML to represent the language, enables the structure of an SLA to be formally defined in XML Schema.

Figure 4.14 illustrates an XML Schema for the language, and Figure 4.15 provides an example SLA for the Stock Quote Service based on this schema. This SLA contains a single SLO comprising of two outcomes and the payments between the organizations for these outcomes. The XML representation provides an XML representation of the Java class structure and data types defined in Figure 4.13. The Java class hierarchy is can be transformed to and from the XML representation by the services on transmission and reception of this data respectively utilizing the XStream library described in Section 4.5.2.3.

4.5.1.2 Agreement Service

The Agreement Service designed in Section 4.4.3.2 is implemented as a Web application, consisting of a set of Java Servlets and Filters, along with any libraries and classes on which these Servlets and Filters are dependent. In accordance with the utilization of a Web application, the REST style to which the design of the Agreement Service adheres is realized utilizing Hypertext Transfer Protocol (HTTP), such that resources are identified by a URI in the form of a Web address, and interaction with the resources of the Agreement Service comprises of a synchronous HTTP request-response pair. The resources support a uniform interface for interaction defined by the HTTP methods, and can utilize the HTTP status codes to denote the semantics of responses. These methods map onto state-centric operations defined in Section 4.4.1 to provide introspection on, and manipulation of the state of resources.

Figure 4.16 provides a table of interactions supported by the Factory resource of the Agreement Service, implemented using HTTP. The factory resource is identified by a URI of the form: `/SLAs?service=services:stock_quote_service`, where `/SLAs` provides a mapping of the request to a specific set of Filters and Servlets associated with the factory resource, and the URI for the specific service for which the SLA is requested is `services:stock_quote_service`. The factory resource supports a single interaction, `GET`. This interaction represents the retrieval of the currently proposed SLA from the Agreement Service. This request contains the data required from the cre-

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="sla">
    <xs:complexType>
      <xs:attribute name="id" type="xs:string"/>
      <xs:attribute name="service" type="xs:anyURI"/>
      <xs:attribute name="user" type="xs:anyURI"/>
      <xs:attribute name="provider" type="xs:anyURI"/>
      <xs:element name="slos">
        <xs:complexType>
          <xs:element name="slo">
            <xs:complexType>
              <xs:attribute name="metric" type="xs:string" use="required"/>
              <xs:sequence>
                <xs:element name="outcome" maxOccurs="unbounded">
                  <xs:complexType>
                    <xs:attribute name="operator" type="xs:string" use="required"/>
                    <xs:attribute name="value" type="xs:decimal" use="required"/>
                    <xs:element name="payment">
                      <xs:complexType>
                        <xs:attribute name="payer" type="xs:string" use="required"/>
                        <xs:attribute name="payee" type="xs:string" use="required"/>
                        <xs:attribute name="value" type="xs:decimal" use="required"/>
                      </xs:complexType>
                    </xs:element>
                  </xs:complexType>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:complexType>
      </xs:element>
      <xs:element name="requests">
        <xs:complexType>
          <xs:element name="request">
            <xs:attribute name="id" type="xs:anyURI"/>
          </xs:element>
        </xs:complexType>
      </xs:element>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Figure 4.14: An XML Schema For SLAs

```
<sla id="123a0123-a12b-12d4-a123-12344321000" service="services:stock_quote_service"
            requester="ABC Investments" provider="XYZ Stockbrokers">
  <slos>
    <slo metric="metrics:response_time">
      <outcome operator="LEQ" value="200">
        <payment payer="ABC Investments" payee="XYZ Stockbrokers" value="0.05"/>
      </outcome>
      <outcome operator="GT" value="200">
        <payment payer="XYZ Stockbrokers" payee="ABC Investments" value="0.07"/>
      </outcome>
    </slo>
  </slos>
  <requests>
    <request id="456b0456-b56g-hg87-jf85-547380987876"/>
  </requests>
</sla>
```

Figure 4.15: An SLA For The Stock Quote Service

Interaction	**Status Code**	**Response Body**
GET	200	Newly created SLA encoded in SLA Hypermedia Format.
	400	-
PUT	405	-
POST	405	-
DELETE	405	-

Figure 4.16: A Table Of Interaction Methods For The Factory Resource

ation of an SLA, namely the organization requesting the SLA, and the service for which the SLA is being requested. As described above, the service is included in the query string, whilst the identity of the organization is provided by the Identity Service. The absence of the necessary data in the `GET` request for the SLA yields a response from the service with a status code of `400` representing a bad request which must be modified in order to successfully request an SLA. In the presence of the necessary data in the request, the response from the `GET` request is contains a status code `200` representing a successful request, and the body of the request contains an SLA in the hypermedia format described in Section 4.5.1.1. The response contains in the metadata, a `content-location` header which contains a URI reference to that particular SLA such that the SLA with that particular ID can be resolved at the negotiation resource. The response from any other interactions requests, such as `PUT` or `POST`, contain a status code representing the absence of support for such an interaction, and an empty request body.

Interaction	**Status Code**	**Response Body**
`GET`	200	Appropriate SLA encoded as XML representation.
	401	-
	404	-
`PUT`	405	-
`POST`	200	Modified SLA encoded as XML representation.
	304	Unmodified SLA encoded as XML representation.
	401	-
	415	Unmodified SLA encoded as XML representation.
`DELETE`	405	-

Figure 4.17: A Table Of Interaction Methods For The Agreement Resource

Figure 4.17 provides a table of interactions supported by the Agreement resources of the Agreement Service, implemented using HTTP. The Agreement resources are identified by a URI of the form: `/SLA?id=123a0123-a12b-12d4-a123-12344321000`, `/SLA` provides a mapping of the request to a specific set of Filters and Servlets associated with the Agreement resources, and where `123a0123-a12b-12d4-a123-12344321000` is a Universal Unique Identifier (UUID) for the specific SLA. The agreement resources support two different interactions: `GET` and `POST`, which correspond to `retrieve()` and `update()`. The `GET` interaction represents the retrieval of the SLA identified by the ID supplied in the query string. In the presence of a valid ID, the request is authenticated and authorized utilizing the Identity Service, such that only an organization in the SLA can retrieve the SLA. An unauthorized organization receives a response from the service containing a status code of `401` and an empty body, while an authorized organization receives the SLA encoded in the SLA XML representation.

The `POST` interaction facilitates the proposal and counter-proposal which constitutes negotiation over the SLOs in the SLA. The organization can interact through `POST` with inclusion of an XML representation of the SLA in the body of the HTTP request, representing the counter-proposal. In an analogous manner to the `GET` interaction, only an authorized organization can perform the proposal or counter-proposal, and a request which is unauthorized will receive a response with the status code `401`. The representation filter performs the transformation of the SLA representation from XML to Java object, and from Java object to XML in the event of another proposal. This transformation utilizes XStream (see Section 4.5.2.3) with appropriate convertors and aliases setup. The inclusion of an invalid XML representation of the SLA in the request body leads to a response from the service with a status code `415`. Such an invalid encoding may be due to the manipulation of the encoding structure such that it no longer conforms to the schema in Figure 4.14. A valid counter-proposal to the service is considered by the service in an accordance with beliefs, objectives and attitudes and either accepted, represented with a status code `201`, or not accepted with a status code `304`. Indeed, as a simplification of the negotiation service, we implement the service to support a "take-it-or-leave-it" style of negotiation, where the service proposes an SLA which is either accepted or not accepted, with no counter-proposal facilitated. The support for such a counter-proposal would be straightforward to implement, simply using the methodology in the SLA Factory to evaluate the expected utility of an existing SLA rather than formulate an optimal SLA, and accepting that counter-proposal if an acceptable level of expected utility is attained. The response from any other interactions requests, `PUT` or `DELETE`, contain a status code representing the absence of support for such an interaction, and an empty request body.

In order to facilitate ease the definition of QoS modelers for each metric, a library of parametric and non-parametric distributions were implemented as classes in Java. These distributions can be utilized by any QoS modeler as a basis for modeling the distribution of a metric. For example, the

empirical data available to the QoS modeler could be utilized to bootstrap a parametric distribution such as an exponential distribution. Of course, such an ability to utilize a parametric distribution is predicated on the knowledge of the organization provisioning the Web service that the values of a certain metric do correspond to a certain distribution. The parametric distributions implemented were the `NormalDistribution`, `ExponentialDistribution` and `TruncatedNormalDistribution`, which require the appropriate parameters to be inputted to them. The non-parametric distribution implemented was the `EmpiricalDistribution` classes, which utilizes empirical data, deriving the probability of values based upon the number of observations in the empirical data within each section of the potential domains of values for the metric.

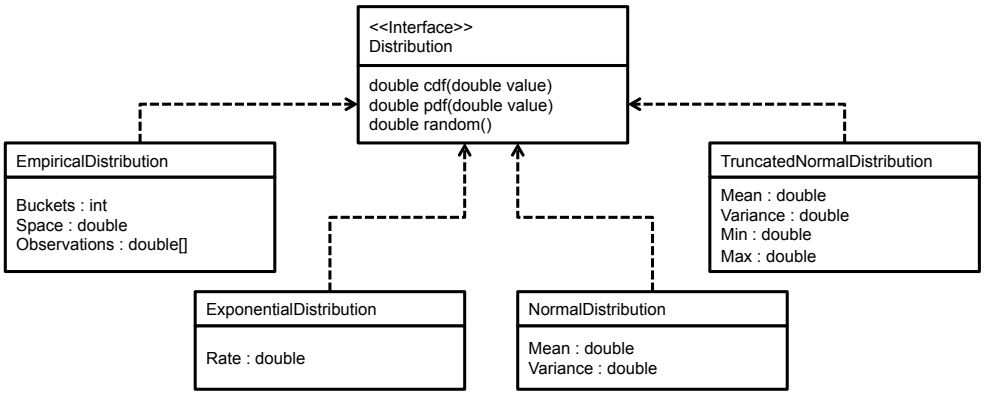

Figure 4.18: A Class Diagram For Probability Distributions

Figure 4.18 provides a class diagram for the different distributions supported by the SLA Factory. The Agreement Service defines a QoS model manager which facilitated the construction of the distribution to utilize in the creation process from empirical data. This distribution manager is based around the utilization of data from a JDBC data-source (in this case the database) to create a distribution. For each metric, a QoS Modeler is defined which defines the metric for which it derives a distribution, and the number of empirical observations it requires from the data source in order to produce a distribution. The `getDistribution()` method is called to create a distribution, including any feedback which may be utilized in the creation of the distribution. This feedback represents auxiliary information, such as system metrics like queue size, which can augment the information utilized in the creation of the distribution, under the assumption that the global context is shared with the Monitor Service. The QoS modeler can be implemented for any of the different types of distribution described above.

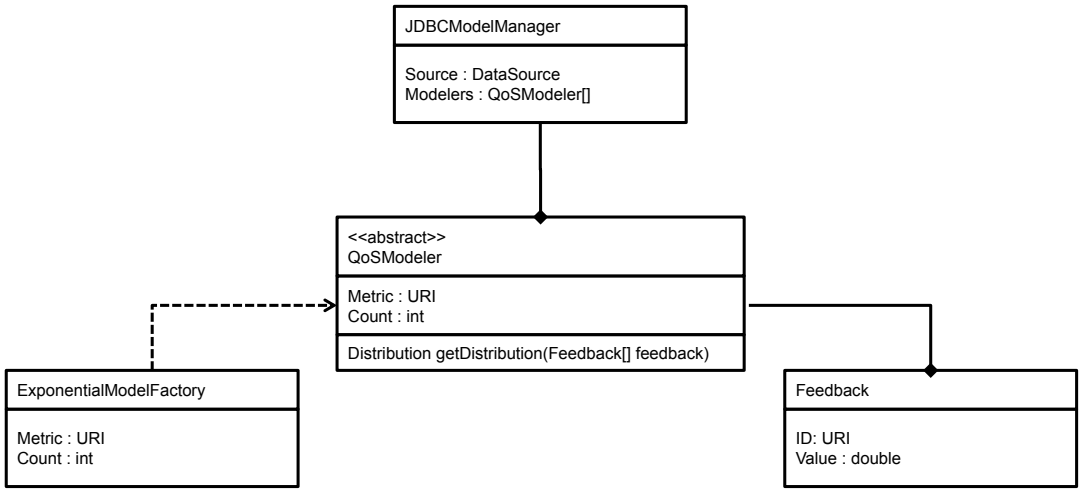

Figure 4.19: A Class Diagram For QoS Model Manager

Figure 4.19 provides a class diagram for the QoS model manager. This model manager is utilized within the creation process by the SLO factory for each metric. This SLO factory combines the beliefs represented by the distribution and the desires of the organization to derive intentions with regard to the SLA. This SLO factory is utilized to create the SLOs for a given metric within an SLA. It is this component which performs the optimization process.

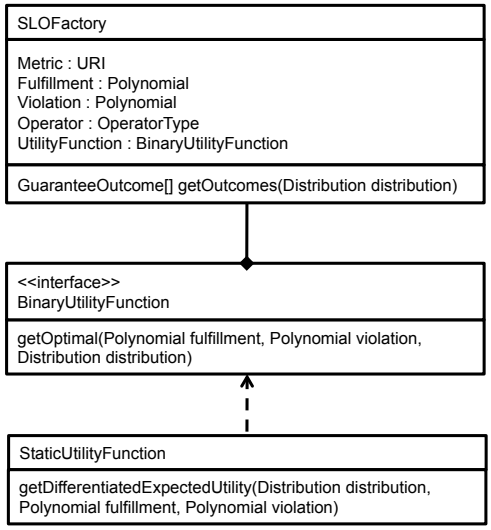

Figure 4.20: A Class Diagram For The SLO Factory

Figure 4.20 provides a class diagram for the SLO Factory. The SLO Factory contains the ID of the metric to which it pertains, the functions which determine the objectives over the success and failure outcomes, in the form of polynomials, an operator which defines the operator utilized to partitioned the domain of values for a metric to create an SLO, and a utility function. The utility

function performs the optimization process for the metric. On the execution of the `getOutcomes()` method, a utility function is constructed from the component parts within the `SLOFactory`, the fulfillment and violation functions, the operator type and the distribution which is passed to the `getOutcomes()` method.

4.5.1.3 Monitor Service

The Monitor Service designed in Section 4.4.3.2 is implemented as a Web application, consisting of a set of Java Servlets and Filters, along with any libraries and classes on which these Servlets and Filters are dependent. In an analogous manner to the Agreement Service, the Monitor Service is realized utilizing Hypertext Transfer Protocol (HTTP), such that the interaction with the resources of the Monitor Service comprises of a synchronous HTTP request-response pair. The resources support a uniform interface for interaction defined by the HTTP methods, and can utilize the HTTP status codes to denote the semantics of responses. These methods map onto state-centric operations defined in Section 4.4.1 to provide introspection on, and manipulation of the state of resources. The outcome resource supports any interaction, as the semantics of the response will be determined not by the Monitor Service but by the Web service to which the request is subsequently dispatched. Therefore, the outcome resource must be able to support any interaction which may be required by any of the Web services it monitors.

The metrics supported by the Monitor Service were bound to the HTTP protocol, such that the status codes of HTTP were utilized to reference the set of responses from the service. Accordingly, the state machine defined in the design of the Monitor Service can have states representing the HTTP status codes. Figure 4.21 shows this concrete incarnation of the state machine. A filter is created for temporal metrics and for boolean metrics, where each filter was initialized with the name of the metric they represented, and the status code which they represented (for boolean metrics). These filters extend those provided by Apache Tomcat (see Section 4.5.2.1), and are placed sequentially within the data flow of the Monitor Service. Figure 4.22 provides a class diagram showing the two different filters created.

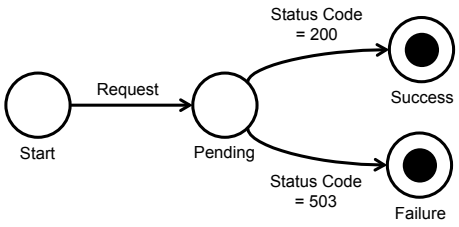

Figure 4.21: A Finite State Automata For Response Time

The temporal metrics were realized using filters which perform a time-stamp of reception of the request from the requester entity, and a time-stamp on the reception of the response from

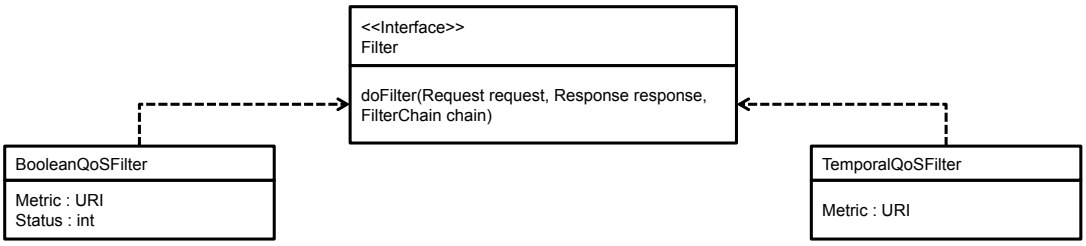

Figure 4.22: A Class Diagram For QoS Filters

the service. The initial time-stamp was stored in context of the request, and the value of the temporal metric was derived on the time-stamp of the response, utilizing the time-stamp stored in the request. This time was then stored in the database in the form: (m, v), and associated with the unique identifier of the request to which it pertained. The boolean metrics were realized using filters which perform a boolean check on the status code in the response from the service. This boolean was then stored in the database in the form: (m, v), where $v \in \{0, 1\}$, and associated with the unique identifier of the request to which it pertained. The Monitor Service was implemented with support for two metrics: `metrics:response_time` and `metrics:unavailable`. These two metrics were implemented as individual Metric Monitors, where the `metrics:response_time` metric was implemented as a temporal metric where the monitor time-stamps requests and responses, and the `metrics:availability` metric was implemented as a boolean metric where the monitor simply checks the status code associated with the response. The status code we associate with the metric was the HTTP status code 503 which has well-defined, precisely defined semantics in the HTTP specification [19].

4.5.2 Implementation Tools

The implementation of the software architecture utilizes a number of different tools. These tools are based upon the Java [200] programming language. Java is ubiquitous in the development of Web services, and consequentially there exists a vast number of tools based upon Java which can be utilized to assist such development. Java programs run within the Java Virtual Machine (JVM), which provides a framework on which these programs can be built.

4.5.2.1 Apache Tomcat

Apache Tomcat [201] is an open source servlet container which provides a Java-based implementation of Java Servlets [202] within a Java-based Web server. The servlet container can contain a number of different Web applications. A Web application is contained within a Web Application Archive (WAR) file. The WAR file contains a context which defines the context path of the Web application. The servlet container directs HTTP requests to a Web application based upon the Request URI

and the context path defined for each Web application. Once the request has been passed to the appropriate Web application, the context will choose the servlet to which the request will be passed for processing, in accordance with the Web application deployment descriptor file located in the WAR file.

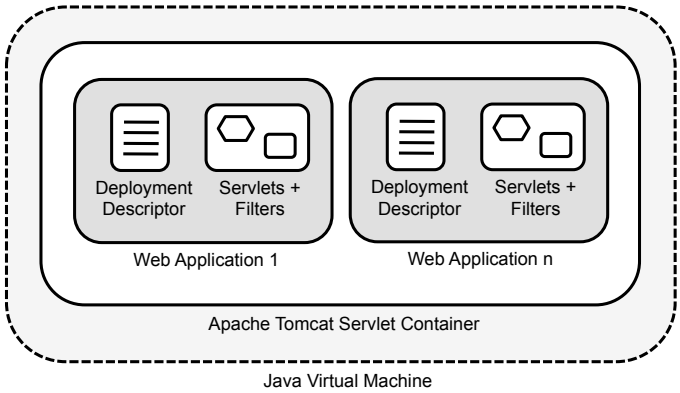

Figure 4.23: An Illustration Of Apache Tomcat

Figure 4.23 illustrates the execution environment of Tomcat, and the Web applications deployed within Tomcat. Servlets are designed to work within a request/response processing model. In a request/response model, a client sends a request message to a server and the server responds by sending back a reply message. Requests can come in the form of any protocol, and the servlets can be implemented to support any protocols, but the Apache Tomcat Servlet Container provides in-built support for HTTP servlets. Accordingly, our implementation will realize the REST architectural style with the HTTP protocol utilizing Java Servlets. These servlets provide the uniform interface to resources, where resources are identified by Uniform Resource Identifier [18], and mapped to functionality through the context of the Web application. The uniform interface comprises of the set of the HTTP methods [19].

Name	**Description**
GET	Retrieve the representation of the resource identified by the URI.
POST	Accept the representation enclosed in the request as a new subordinate of the resource identified by the URI.
PUT	Store the enclosed representation under the URI.
DELETE	Delete the resource identified by the URI.

Figure 4.24: A Table Of HTTP Methods

Figure 4.24 summarizes the methods utilized in the implementation. These methods provide introspection on, and manipulation of resources utilizing their representations. A request to a certain resource, through the uniform interface, passes through a chain of filters before being ultimately processed by a servlet, which performs some business logic on the request. The pipe-and-filter style of the design is realized using this chain of filters and servlet on each request and response pair. This can include the pre-processing of request prior to the servlet, and the post-processing of responses from the servlet. These pipes and filters are combined to form a component within the deployment descriptor, which maps a URI (a resource) to a chain of pipes and filters which comprise the component. The responses from requests utilize the HTTP status codes [19] to reflect the semantics of the response, such as fulfillment or violation of the request. Figure 4.25 summarizes some of the broad categories of status codes utilized in the implementation.

Status Code	Description
1xx	Request received, continuing process.
2xx	Action successfully received, understood and accepted.
3xx	Further action required to complete request.
4xx	Request contains bad syntax or cannot be fulfilled.
5xx	Server failed to fulfill valid request.

Figure 4.25: A Table Of HTTP Status Codes

4.5.2.2 Apache Derby

Apache Derby [203] is an open source Relational Database Management System (RDBMS) which provides support for Structured Query Language (SQL) through a Java-based implementation. The Derby database engine enables connections through the Java Database Connectivity (JDBC) application programming interface utilized in Java applications to provide uniform methods for introspection on, and manipulation of a relational database. Derby supports the creation of multiple databases within the same database engine, and a Derby system comprises of the database engine along with the environment in which the engine runs. The system comprises of a system directory which contains any persistent system-wide configuration parameters, or properties, specific to that system in a properties file and zero or more databases. Each database within the system is contained in a subdirectory, which has the same name as the database.

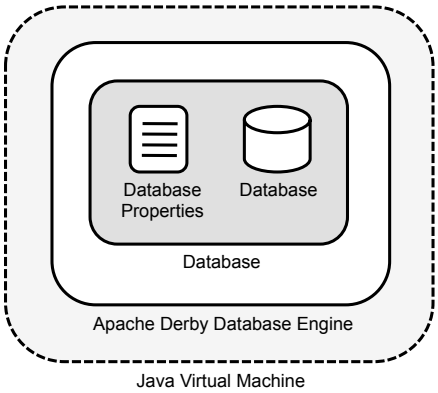

Figure 4.26: An Illustration Of Apache Derby

Figure 4.26 illustrates the execution environment of Derby. There is support for two different environments within which the database can reside: embedded and server. The embedded environment enables the database to exist within a local file system, such that connections to the database are internal to the system on which the database resides. The Derby database engine runs within the same Java Virtual Machine (JVM) as the Java application which utilizes the database. This enables persistent storage to be achieved within the application without the complexity and overhead of a separate database process. The server environment enables the database to exist on a remote server, such that connections to the database are external to the system on which the database resides. The Derby Network Server embeds Derby and handles connections to the database from network client in different JVMs on the same machine or on remote machines. We utilize the embedded environment for the purposes of simplicity since it enables the persistent storage of data to be performed in the same JVM as the servlet container. This provides an elegant encapsulation of functionality, with the trade-off arising from the increased load on the resources of the JVM. The embedded database is placed within the Web application and accessed from within servlets utilizing JDBC connections. This means that the Web application can be easily transferred across machines with the servlet container installed without fear of configuration from changes in the addressing of the database.

4.5.2.3 XStream

XStream [204] is a library for the serialization of Java objects to XML, and the parsing of XML to Java objects, based upon the Xerces XML Parser [205]. The library is based around the notion of a converter which takes a Java object and converts the object to a serialized XML representation, or takes a serialized XML representation and converts the representation to a Java object. There exists a large number of converters for standard Java types such as Integers and Booleans. Additionally, the library facilitates the creation of custom converters which can stipulate with greater precision how custom classes are converted to and from XML. The serialization to and from XML can be

customized further utilizing aliases, which enable attribute within a Java object to be displayed as attributes in the XML serialization. An XStream object is created to perform the serialization and/or parsing, and the aliases and custom converters are added to the configuration of this object.

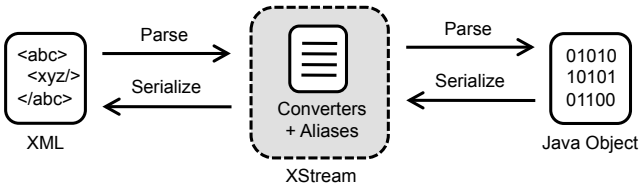

Figure 4.27: An Illustration Of XStream

Figure 4.27 illustrates the process of conversion between XML representations and Java objects using XStream. The operations `xstream.toXML(myObject)` or a `xstream.fromXML(myXML)` are performed on the Java object and XML representation respectively to perform conversion.

4.6 Deployment

The deployment of a software architecture provides a description of the placement of architectural components onto computational resources. The Web services which represent the architectural pre-requisites were included in the deployment in an indicative manner. These Web services were not invoked during the experimentation which utilized these deployments (see Chapter 5) for reasons of augmented cost and complexity. The encapsulation of the Agreement Service and Monitor Service within Web Application Archive (WAR) files facilitates their ease of deployment within any Web application container, such as Apache Tomcat 4.5.2.1. This provides flexibility in which organizations can utilized the Web services since they exhibit portability and are not tied to any specific hardware or software environment.

In the next two subsections, we describe the deployment of the software architecture for utilization with Web services provisioned on two different computational infrastructures: Grid and Cloud.

4.6.1 Grid Deployment

The Grid deployment places the Web service for which the SLAs are created, the Stock Quote Service, on a Grid computing infrastructure. This deployment is utilized in the experimental procedure of the empirical study described in Chapter 5. This Grid computing infrastructure is located at Newcastle University, and consists of a set of 20 homogenous Linux machines. An indicative, dummy Stock Quote Service is deployed on each of these machines, which receives a request, waits for a randomly determined time period and then returns a response. The Web service adheres to

the REST architectural style, and supports a single operation for introspection on state, namely GET. The Agreement Service and Monitor Service are deployed on a Windows machine within the Local Area Network (LAN) of Newcastle University, and are assumed to be provided by a third party organization, DEF Agreements. The Monitor Service is configured with the Grid Service Manager specific to the utilization of a Grid by the Web service being monitored. The requester performs interactions with the Agreement Service and Monitor Service over the World Wide Web. Additionally, the Agreement Service and Monitor Service perform interactions with the Identity Service of GHI Identity, and the Payment Service of JKL Payments over the World Wide Web.

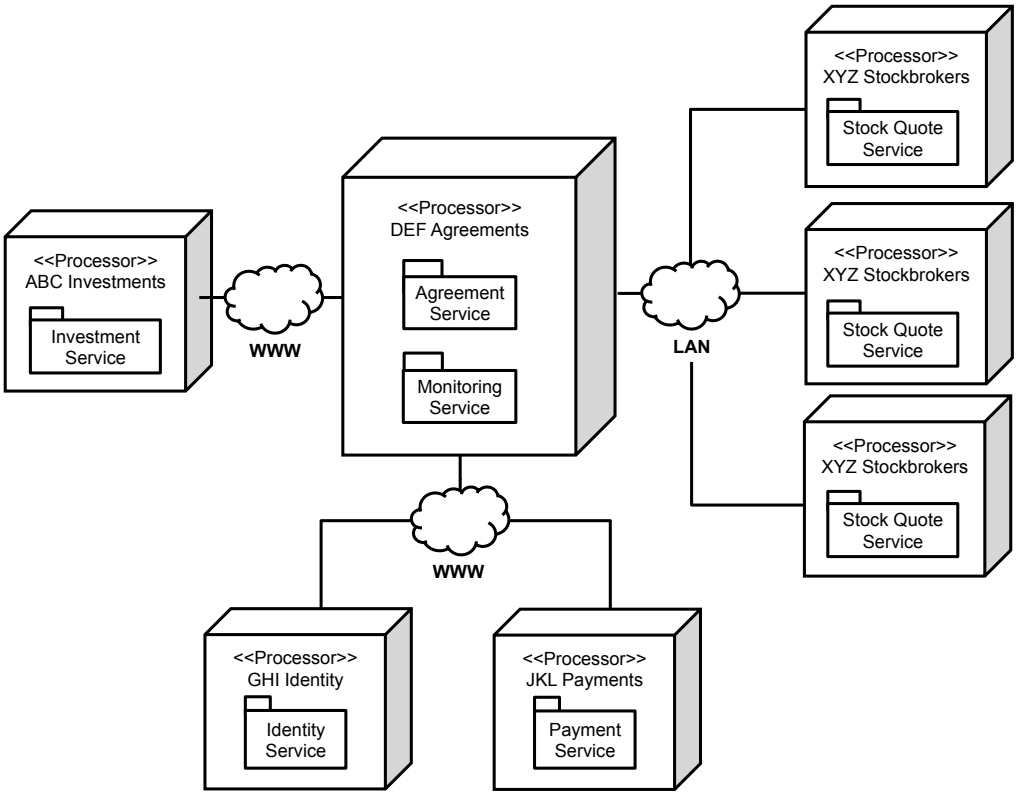

Figure 4.28: A Deployment Diagram For A Grid-Based Web Service

This deployment assumes that the Agreement and Monitor Services are provisioned by DEF Agreements, which is a TTP. This organization is assumed to reflect the desires of the provider with regard to optimality of SLAs, but reflect objectivity with regard to the monitoring and enforcement of these SLAs. Alternatively, the Agreement and Monitor services could be deployed at different third party organizations, such as a dedicated broker for SLAs and a dedicated monitor for SLAs. Such a deployment may increase the trust of the requester in the third party organization, due to the absence of the potential conflict of objectives which may arise from reflecting the desires of the provider with certain functionality and not with other functionality.

4.6.2 Cloud Deployment

The Cloud deployment placed the Web service for which the SLAs are created on a Cloud computing infrastructure. This Cloud computing infrastructure was provided by Amazon [105]. The Simple Stock Quote Service (S3) [206] is deployed on this cloud by Amazon, which enables the storage and retrieval of data objects. The S3 Web service adheres to the REST architectural style, and supports operations for introspection on state, representing retrieval of a data object, and manipulation of state, representing storage of a data object.

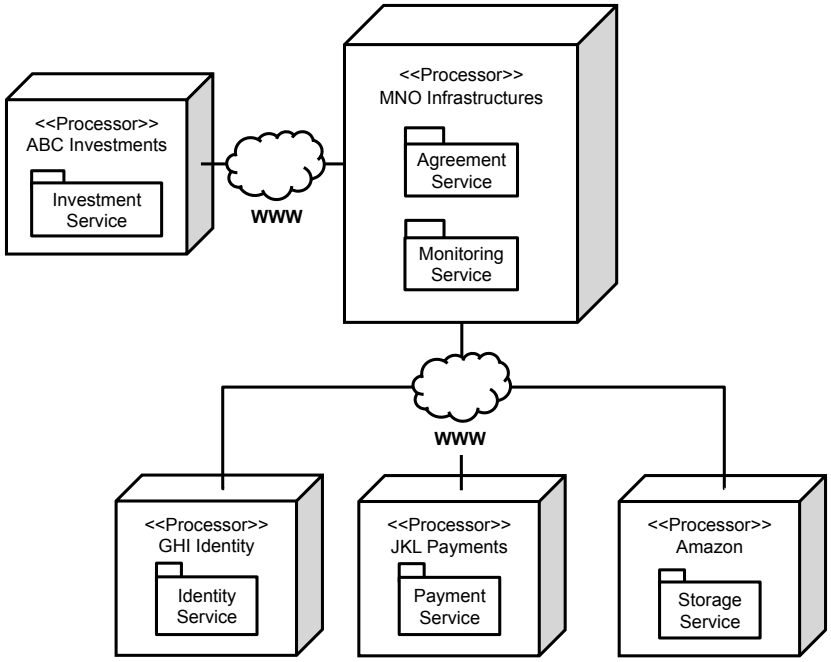

Figure 4.29: A Deployment Diagram For A Cloud-Based Web Service

In an analogous manner to the Grid deployment, the Agreement Service and Monitor Service is deployed on a Windows machine within the Local Area Network (LAN) of Newcastle University, and are assumed to be provided by MNO Infrastructures. The Monitor Service is configured with the Cloud Service Manager specific to the utilization of a Cloud by the Web service being monitored. The requester performs interactions with the Agreement Service and Monitor Service over the World Wide Web. Additionally, the Agreement Service and Monitor Service perform interactions with the identity, payment and Stock Quote Services over the World Wide Web. This deployment assumes that the Agreement and Monitor Services are provisioned by a TTP. In contrast to the Grid deployment, the organization provisioning the Agreement and Monitor Services is assumed to reflect its own desires and objectives with regard to the optimality of SLAs. MNO Infrastructures is assumed to provision a Web service which provides the functionality of a broker for requesters, reselling the Stock Quote Service provisioned by Amazon. This broker can provide additional QoS constraints

to those provided by Amazon, trading on beliefs acquired from monitoring of interaction with the Stock Quote Service, and the creation of SLAs which are optimal with regard to these beliefs and organizational objectives.

4.7 Evaluation

The software architecture defined in this chapter provides an instantiation of an environment for SPRs, as described by the framework in Chapter 3. This software architecture can be evaluated against the set of functional and non-functional requirements which were defined for the software architecture in Section 4.2 in order to determine the efficacy with which the software architecture has been designed and implemented in accordance with these requirements. This efficacy is then demonstrated in Chapter 5 which utilizes the software architecture to perform an empirical study of uncertainty for the provider in the creation of SLAs as contingent contracts.

The functional requirements stipulated that the software architecture must provide a structured language for the representation of SLAs as contingent contracts. Such a structured language is defined within the software architecture, and utilizes HTTP status codes in order to provide verifiable and enforceable outcomes, whose semantics are well-defined and understood by organizations. The language facilitates the definition of multiple SLOs which partition the space of possible metric values into two or more contingencies. The partitioning of the domain in this manner can avoid many instances of incomplete contracts, where contingencies arise which are not included in the contract. These contingencies can then be associated with a payment between the requester and provider. Each request to a Web service is assigned a unique identifier such that the outcome of each request is verifiable, and the appropriate contingency can be enforced for that request. This language was utilized in the Agreement Service to create and negotiate SLAs which are optimal with regard to certain organizational objectives, and therefore fulfill the second functional requirement. This Web service is instantiated with certain organizational objectives, in the form of a utility function, and a QoS modeler to facilitate the derivation of beliefs over QoS. An optimization problem is then solved by the Web service for each metric in order to derive an SLA whose SLOs are optimal with regard to the organizational objectives and the beliefs of the organization.

The monitoring of these contingencies was realized by the Monitor Service, which intercepts requests to a Web service and stores the outcome of each request to facilitate the enactment of the appropriate contingencies with an SLA based on that outcome. In essence, the Monitor Service performs the centralized coordination of the interactions within an SPR. This involves the verification of the QoS experienced, and the enactment of contingencies within a contingent contract based upon this QoS. Clearly, such coordination must be performed by a trustworthy organization, otherwise the organization could simply choose not to enact a certain Web service when it is contrary to

organizational objectives. For example, if the provider provisions the Monitor Service he may detect that the QoS is such that a monetary payment must be made to the requester, but may choose not to enact the Payment Service to avoid payment. Therefore, the Monitor Service encapsulates coordination functionality in addition to the detection of the QoS. Such functionality could be stripped out of the Monitor Service, but would then require provisioning from an appropriately trustworthy organization, otherwise the SLA would lack the credible threat it requires to control the provider. The communication with Web services by the Monitor Service in order to coordinate the SPR is subject to the constraints of the Communication Network at the Communication Layer, namely the WWW. These constraints apply not only to the invocation of the Web service at the Service Layer, such as the Stock Quote Service, but also the invocation of Web services such as the Identity Service or the Payment Service. Accordingly, the invocation of such services is subject to the reliability and latency properties of the communication network. Such properties may prevent the correct invocation of certain Web services at the Platform Layer and may therefore prevent the controls at the Platform Layer from being enforced. For example, the Monitor Service may enact the Payment Service to make a payment between organizations for the experience of a certain QoS. The Monitor Service may invoke the Payment Service, but the communication network may lose or corrupt one or more of the messages which comprise the invocation. Therefore, whilst the Web services may be provisioned in a trustworthy manner, the coordinated enactment of the Web services may not be trustworthy due to the constraints of the communications network. This provides a concrete illustration of the framework layers, which dictate that as we move upward through the layers, the controls at one layer are subject to the controls of the layer below. Such issues of reliability in the communication network illustrate this point in a lucid manner.

The non-functional requirements stipulated that the structured language for SLAs should be sufficiently simple to enable straightforward design and evaluation of SLAs with regard to organizational objectives. The language demonstrates such simplicity by including only the minimal detail required to commit organizations to certain obligations, and to enforce these obligations through payments. Each metric can be simply defined utilizing HTTP status codes, and a notion of time, and the fulfillment or violation (or more granular notions of fulfillment or violation) of SLOs based on these metrics can be simply defined through the partitioning of the space of values into two or more partitions. The SLAs represented by this structured language, and created by the Agreement Service, are highly adaptable to changes in organizational objectives. An organization is simply required to redefine the utility function for a certain metric, which defines the monetary payment yielded by the organization for each value of the metric, considering factors such as management costs. This redefinition is performed through straightforward manipulation of a configuration file within the Agreement Service. The creation of SLAs, and the monitoring of the QoS for enforcement of these SLAs can be easily extended to different Web services and different metrics. For any given

Web service, the Agreement Service must have a unique identifier for that Web service, a definition of the organizational objectives with regard to the metric, and a QoS modeler for that metric by which the beliefs over the values of the metric can be derived. The Agreement Service can then create SLAs for any given Web service, based on these objectives and beliefs. The metrics included in the SLAs for each Web service can be manipulated such that different Web services support different metrics. The Monitor Service retains a list of Web service for which monitoring is supported, and forwards requests and responses to and from these Web services after the performance of the appropriate monitoring functionality. Clearly, in order for an SLA which is created by an Agreement Service to be enforced, the Monitor Service which provides the monitoring and enforcement must support monitoring for the Web service and metrics included within the SLA.

4.8 Conclusion

This chapter has addressed the problem of uncertainty in QoS experienced by the requester and provider of a Web service. The chapter presents the design and implementation of a software architecture which provides an environment for SPRs. The chapter has two main contributions. Firstly, a structured language for the representation of SLAs as contingent contracts, whose contingencies are defined over the QoS experienced by the requester. Secondly, a practical methodology for the creation of SLAs as contingent contracts, whose contingencies are optimal with regard to any organizational objectives, in the presence of uncertainty in the QoS. In the next chapter, the efficacy of the software architecture is demonstrated through its utilization to carry out an empirical study of uncertainty in QoS experienced by the provider of a Web service in an SPR.

Chapter 5

An Empirical Study Of Uncertainty In Service Provisioning Relationships

This chapter addresses the problem of uncertainty in QoS experienced by the provider of a Web service. The chapter presents an empirical study of uncertainty in SPRs carried out utilizing the software architecture presented in Chapter 4. The main contribution of the chapter is a theoretical description of a methodology for the creation of SLAs as contingent contracts, whose contingencies are optimal with regard to any organizational objectives, in the presence of uncertainty in the QoS. Such an approach differs from previous work, which does not provide a generic process for the creation of SLAs for any QoS metric of any Web service, given organizational objectives. The value of such an approach is to provide a formal process for the creation of optimal SLAs, such that organizations no longer utilize processes based on ad-hoc, sub-optimal reasoning which can be inconsistent with organizational objectives.

The remainder of the chapter is structured as follows. Section 5.1 introduces the problem domain of uncertainty which is addressed in this chapter. Section 5.2 formulates a set of requirements for the empirical study to investigate the problem of uncertainty in the model of the QoS. Section 5.3 lists the assumptions of the empirical study. Section 5.4 discusses the methodology which is utilized to perform the empirical study. Section 5.5 describes the experimental procedures performed to obtain the results of the empirical study. Section 5.6 presents the empirical results obtained from experimentation and discusses their significance. Finally, the chapter is briefly summarized and concluded in Section 5.7.

5.1 Introduction

Uncertainty in QoS can be experienced by the provider of a Web service. The provider of a Web service can be uncertain of his ability to provision a Web service with a certain QoS. This uncertainty

arises from a lack of complete control over, and complete information pertaining to, the computing infrastructure on which the Web service is provisioned. The infrastructure can be prone to resource failures and fluctuations in usage patterns, introducing variability in resource capacity and QoS. Such uncertainty leaves the provider vulnerable to the provision of a Web service with a QoS which does not meet the requirements of a requester, leading to negative effects on the fulfillment of organizational objectives. This uncertainty can endanger the economic viability of SPRs, and mitigate any increases in organizational efficiency for the provider. The organization may prefer not to participate in SPRs given the presence of such uncertainty, due to the negative effects on the fulfillment of organizational objectives from provision of the Web service with a certain QoS.

The provider can address uncertainty in the QoS of a Web service with management. Management involves the "monitoring, controlling and reporting of, service qualities and service usage" [28] by an organization such that a Web service "fulfills the requirements of both owners and users of the system" [20]. Management mechanisms are those mechanisms utilized by the provider to reduce the uncertainty in the QoS through the imposition of control over the computing infrastructure on which the Web service is provisioned, or through information pertaining to the computing infrastructure. The resource capacity of the computing infrastructure can be subject to failures and fluctuations in service usage which reside outside the control of the provider and influence the QoS. The absence of control of the provider over such failures and fluctuations, dictates that the provider must utilize information to reduce such uncertainty. This information encapsulates the capabilities of the provider with regard to the QoS of a Web service and is represented by a model of QoS. This model is subject to the bounded-rationality of the provider which can introduce uncertainty into the model, where the ability to provision the Web service with a certain QoS is either over-estimated or under-estimated. Additionally, the derivation of the model is subject to costs through the consumption of resources leading to a cost-benefit trade-off for management mechanisms.

In accordance with Chapter 4, contingent contracts may be utilized in order to reduce the uncertainty of the requester in the QoS of a Web service. The management of the Web service by the provider be consistent with the contingent contracts which determine his obligations in an SPR. The participation of the provider in an SPR controlled by a contingent contract requires that the provider must have an accurate model of the QoS of the Web service. The absence of such a model can lead to participation in SPRs which are contrary to organizational objectives, with the provider prone to the efficiency losses of over-estimation, such as negative monetary payments, or under-estimation of QoS, such as organizational slack. The inability of the provider to accurately determine the uncertainty over the QoS of a Web service leads to an inability to determine the uncertainty over which contingency of the contract will be enacted, and therefore the contractual terms to which the provider will be obligated. Given that such contracts can shift some or all of the uncertainty in QoS onto the provider, the provider must be able to effectively quantify such uncertainty with

appropriate management mechanisms, such that contingent contracts can be created or participated in by the provider which are consistent with this uncertainty. Accordingly, this chapter presents a theoretical methodology for the creation of SLAs as contingent contracts, whose contingencies are optimal with regard to certain organizational objectives. This methodology is utilized to perform simulation in order to investigate the impact of monitoring cost on organizational objectives, and the practical realization of this methodology within the software architecture presented in Chapter 4 is utilized to perform investigation into the cost and benefit of QoS modeling methods.

5.2 Requirements

In this section, we describe the requirements of the empirical study of uncertainty in SLAs. These requirements motivate the experimental procedures and empirical results obtained within this empirical study.

1. **A methodology for the creation of optimal SLAs**
   The methodology must enable the creation of SLAs whose contingencies, or SLOs, are optimal with regard to the objectives of the provider. These objectives are assumed to be revenue on a per-SLA basis, such that the SLA contains SLOs which generate optimal revenue for the SLA given the presence of uncertainty in the metrics over which the SLOs are defined.

2. **An empirical study of the negative consequences of uncertainty in QoS, the costs of QoS modeling methods and benefit of different QoS modeling methods**
   The study must demonstrate the impact of epistemic uncertainty in the QoS, arising from the QoS modeling methods utilized by the provider, on the ability to create SLAs whose SLOs are optimal with regard to the objectives of a provider within the SPR. Additionally, the study must demonstrate the different types of costs, arising from the QoS modeling methods utilized by the provider, on the ability to create SLAs whose SLOs are optimal with regard to the objectives of the provider within the SPR. Finally, the study must demonstrate the varied benefit of different QoS modeling methods under different service usage patterns, in terms of the reduction of uncertainty and the effect on the objectives of the provider within the SPR.

5.3 Study Assumptions

The empirical study presented in this chapter is based upon a set of assumptions which include those defined by the framework in Chapter 3, those defined by the architecture in Chapter 4 and the following additional assumptions:

1. **The SLA is optimal for the objectives of the provider**

The SLA is assumed to the formulated to be optimal with regard to the objectives of provider of the Web service, given the beliefs, or QoS model, of the provider obtained from management of the computing infrastructure on which the Web service is provisioned.

2. **The SLA is adapted in accordance with the QoS model**
   The SLA is assumed to be adapted in accordance with the QoS model. This is in contrast to much of the previous work which assumes the SLA to be static and controls the computing infrastructure to fulfill the SLOs of that SLA. The adaption of SLAs enables the provider to quickly adapt to circumstances outside of his control such as resource failures or fluctuations in service usage, over a certain period of time. Such adaption may be utilized as a short-term measure whilst the computing infrastructure is adapted to changes in resource capacity, such as the addition of more resources.

5.4 Methodology

In this section, we describe the methodology which will form the basis of the empirical study. This methodology includes a novel methodology for optimal SLAs under uncertainty in the QoS, a description of QoS modeling methods and the presentation of four different QoS modeling methods which will be utilized for illustrative purposes in the experimental procedure.

5.4.1 Methodology For Optimal SLAs

The methodology we present can facilitate the creation of SLAs which are optimal with regard to the objective function of any organization. Therefore, the SLA could be optimal with regard to the objective function of the requester, the provider or a third party involved in the creation of SLAs. For example, an industry body may be a third party which creates SLAs for SPRs which are efficient and/or equitable. In this study, we assume that the optimality of the SLA is in terms of the objectives of the provider and manifested through factors such as profit or revenue. The methodology could be easily adapted to represent the objective function of other organizations in the SPR such as the requester or a third party.

The provider creates those SLAs, or intentions, which unilateral maximize his desires, given his beliefs over the QoS. The notion of optimality for the provider is represented by maximization of expected utility, where utility can represent business metrics such as revenue or profit. Alternative objectives may include the minimization of losses or fulfillment of a given threshold for revenue on a per SLA basis. These alternative metrics could be incorporated through the manipulation of the terms within the expected utility function and/or the imposition of auxiliary constraints on optimization problem. Given that the SLAs created by the methodology are optimal with regard

to beliefs, the SLAs yielded by the methodology will change with changes in beliefs over time. This enables the SLAs to adapt to changes in beliefs, and remain optimal with regard to the beliefs and objectives of the provider.

The SLA is created at the negotiation stage of the SPR, and follows a simple negotiation process. The provider creates SLAs which are unilaterally optimal with regard to his objectives, and then offers these SLAs to requesters of the Web service for acceptance or rejection. There is no consideration of a counter-proposal from the requester, but the work could be easily extended to consider the proposal and counter-proposal of SLAs to yield an SLA which is mutually acceptable, but not necessarily optimal, with regard to the objectives of both requester and provider. Accordingly, the negotiation and settlement stage of the SPR between the requester and provider comprise of the following steps:

1. The requester requests a SLA for a Web service.
2. The provider formulates an SLA utilizing the methodology for optimal SLA,s and a model of the QoS derived from the chosen QoS modeling method, and responds to the requester with the SLA.
3. The requester accepts the SLA through submission of a request for the service.
4. The provider services the request for the service, and responds to the requester with the service result.

The SLA which is created by the methodology in accordance with the beliefs and objectives of the provider, has the following structure:

Definition 1 (SLA) *A set of SLOs,* $\{s_1, \ldots, s_n\}$.

For ease of analysis we restrict SLAs to containing a single SLO, such that the SLO and SLA can be referred to in an analogous manner. An SLO is defined to contain monetary payments for the experience of two levels of QoS. These two level of QoS are two different outcomes which can emerge in the SPR.

Definition 2 (Outcome) *A pair,* $o = (m, v)$*, where* m *is some chosen quality metric and* $v \in \mathbb{R}$ *is an assigned value for* m.

Accordingly, there exists a binary outcome set for an SLO, $\mathcal{O} = \{o, \neg o\}$, where o represents the service usage with a certain QoS, and $\neg o$ represents service usage without a certain QoS. This outcome set could be easily augmented to represent varied levels of fulfillment and violation, but for reasons of brevity we restrict our discussion to a binary outcome set.

Definition 3 (SLO) *A pair,* $s = ((o, g_m(v)), (\neg o, h_m(v)))$ *were* o *and* $\neg o$ *are outcomes and* $g_m(v)$, $h_m(v) \in \mathbb{R}$ *are the monetary payments for these outcomes.*

For example, the SLO in the SLA presented in Figure 4.15 has the form: `((response_time, 200), 0.05), (¬(response_time, 200), -0.07)`, where 200 milliseconds is an upper bound on the value of the response time metric, and where the monetary payments for the outcomes to the provider are 0.05 and -0.07 respectively. Under the assumption that a metric is defined over a continuum of values, where the continuum represents the space of possible outcomes, the definition of an SLO represents the partitioning of the outcome space, $\mathbb{R}$, into two sub-domains: $V^+_{m,v} \subset \mathbb{R}$ and $V^-_{m,v} = \mathbb{R} \setminus V^+_{m,v}$. The domain of values, $V^+_{m,v}$ represents the domain over which the SLO is deemed to be fulfilled, and the service is supplied with the QoS. The domain of values, $V^-_{m,v}$, represents the domain over which the SLO is deemed to be violated, and the service is supplied without the QoS. In order to model the uncertainty in the QoS for a given metric m, a continuous random variable V_m is defined to represent the value v, with an associated probability density function, $f_m(v)$. This random variable models the aleatory uncertainty of the metric arising from the computing infrastructure.

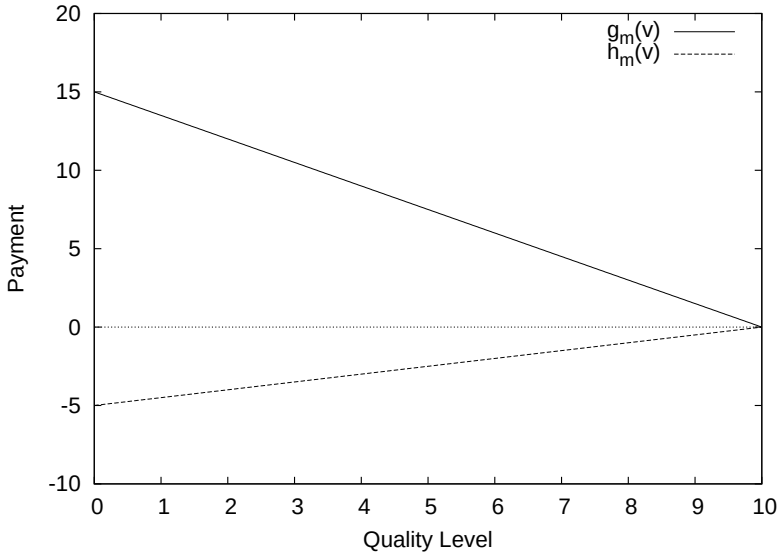

Figure 5.1: An Graph Of Example Payment Functions
$g_m(v) = -1.5v + 15$, $h_m(v) = 0.5v - 5$

The monetary payments associated with the outcomes are determined by a pair of payment functions, $g_m(v)$ and $h_m(v)$ (see Figure 5.1). The payment function $g_m(v)$ determines the payment from the requester to the provider for the fulfillment of the SLO, whilst the payment function $h_m(v)$ determines the payment from the requester to the provider for the violation of the SLO. For

simplicity, we utilize simple linear payment functions to explicate our methodology, yet different payment functions can be utilized to represent different objectives of the provider.

In order to simplify the definition of the payment functions and utilize easily manageable terms, we normalize the values of any given metric to be defined over the domain of values $[0, 10]$ rather than $V^+_{m,v} \cup V^-_{m,v}$, where v_{min} represents the lower bound of this domain and v_{max} represents the upper bound of this domain[1]. For example, the response time metric may be defined over $[0, 10]$ rather than $[0, 5000]$. Alternatively, the payment functions could be defined over the actual domain of the metric through appropriate manipulation of the coefficient in the payment functions and the constant terms such that they realistically reflect the objectives of an organization. Let v'_{max} represent the normalized upper bound of the domain, and v' represent the normalized value of v, the following normalization formula is utilized:

$$v' = \frac{v - v_{min}}{v_{max} - v_{min}} \cdot v'_{max} \tag{5.1}$$

Given our definition of a random variable to represent the value of a metric, and payment functions to define the monetary payments for different values of the metric over the outcome space, the expected utility of a provider for a given contingent SLA can be defined by:

$$EU(m, v) = \int_{x \in V^+_{m,v}} f_m(x)dx \cdot g_m(v) + \int_{x \in V^-_{m,v}} f_m(x)dx \cdot h_m(v) \tag{5.2}$$

The expected utility function, $EU(m, v)$, enables reasoning over the SLA with explicit consideration of the uncertainty in the QoS. Given any value, v, for metric m, the expected utility can be calculated utilizing Equation 5.2. The selection of an optimal SLA is a case of selecting the appropriate partitioning of the outcome space into two outcomes, in accordance with the objectives, and will utilize the optimal QoS, $q^* = (m, v^*)$, where $v = v^*$ is such that:

$$\frac{dEU(m, v)}{dv} = 0 \tag{5.3}$$

Figure 5.2 illustrates the derivation of the optimal QoS, q^*, for selected payment functions and uncertainty, yielding optimal expected utility, EU^*. This optimal QoS represents a local optima rather than a global optima, and accordingly is optimal only over some defined domain of values for the metric. The optimal QoS will vary with different payment functions and density functions (uncertainty in QoS), which change the terms within the expected utility function. In presence of a perfectly accurate model of the aleatory uncertainty in the QoS, such as SLA will maximize the expected utility of the provider. Accordingly, the provider can adapt the SLA to the changes in the QoS such that expected utility is always optimized.

[1]For a temporal metric, such as response time, a low QoS is better than a high QoS, since a higher normalized value represents a higher response time.

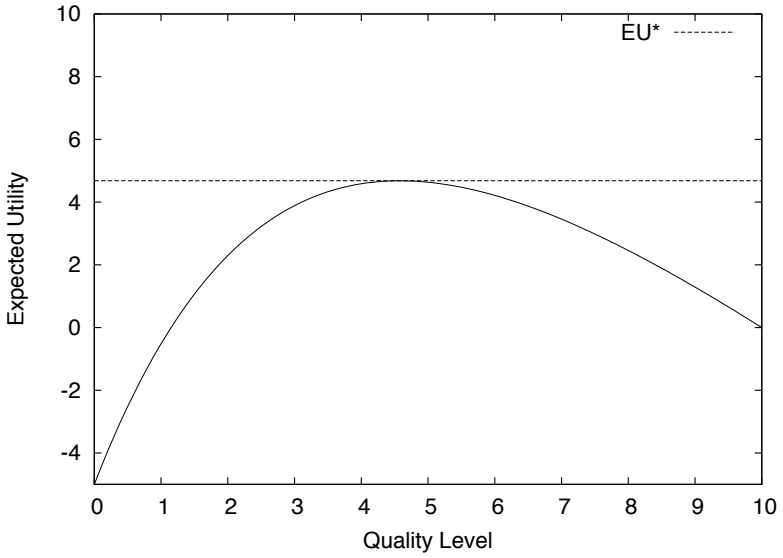

Figure 5.2: A Graph Of Quality Level vs Expected Utility
$g_m(v) = -1.5v + 15$, $h_m(v) = 0.5v - 5$, $f_m(v) = exp(\lambda = 0.25)$

The SLA formulated in accordance with the objectives of the provider does not necessarily provide an appropriate level of trust for the requester. The trust is derived from the extent to which the payment functions shift the uncertainty from the requester to the provider such that the disposition of the requester is sufficient to participate in the SPR. The requester must perceive an expectation of objectives which exceeds his participation constraint from the SPR defined by the SLA. In the absence of information with regard to the provider and the QoS on which to base trust, the trust must arise from constraints imposed at the Platform Layer. Under the assumption that the provider is self-interested, the SLA must be such that the participation constraints of a range of requesters (distinguished by their utility functions) would be fulfilled in the absence of information relating to the provider. The expected values of objectives required for participation may differ amongst different requesters with different beliefs and objectives. For a given requester we denote the value by, $\overline{u}$ and define the participation constraint below:

$$EU_c(m, v) \geq \overline{u}, \forall v \in \mathbb{R} \tag{5.4}$$

Indeed, the payment functions shown in Figure 5.1 provide a greater than money-back guarantee, and represents the fact that the failure of the Web service to fulfill a certain QoS is costly for the requester. The requester is assumed to be at least indifferent between the payment received in the event that the QoS is violated and service usage in accordance with a certain QoS, such that the

payment is an adequate substitute for the service. Therefore, in the constraint above, we assume that $\overline{u} = 0$. Such a transfer may vary on the nature of the service, such that mission-critical services have a high payment due to the monetary impact of violated QoS. Clearly, such SLAs shift all the uncertainty of the QoS onto the provider, with the requester able to verify (with trust in the enforcement of the SLA) that regardless of the outcome, his participation constraint will be fulfilled.

5.4.2 Methodologies For QoS Modeling

The methodology defined in this section facilitates the formation of SLAs under uncertainty in the QoS. It has been assumed that the random variable V_m represents a perfectly accurate model of the aleatory uncertainty in the QoS. Such a model will, though, be subject to epistemic uncertainty due to the bounded capacity of the provider to enumerate and quantify all influencing factors on the computing infrastructure, and therefore to accurately model the aleatory uncertainty. In the next section, we introduce four different QoS modeling methods to illustrate different QoS modeling methods by which the model of the QoS, V_m, can be derived by the provider for subsequent utilization in the methodology. In this section, we discuss the utilization of QoS modeling methods to derive a model of the aleatory uncertainty in the QoS.

The QoS modeling methods monitor the computing infrastructure to derive a model (in the form of a random variable V_m), which is utilized in the methodology described in Section 5.4.1. These QoS modeling methods utilize different perceptions to derive a model of the QoS. The models seek to most accurately model the aleatory uncertainty in the QoS, and therefore to minimize the epistemic uncertainty. The provider is subject to the aleatory uncertainty of the QoS, and to the epistemic uncertainty of the QoS modeling methods which model this aleatory uncertainty when utilizing the methodology in Section 5.4.1.

A QoS modeling method generates an estimator, $\overline{f_m}(v)$, from a set of empirical observations of the QoS, such as previous values of experience metrics and current values of system metrics, to represent a model of the QoS. In the case that $f_m(v) = \overline{f_m}(v)$, no epistemic uncertainty is present and a completely accurate estimator of the aleatory uncertainty is generated by the method. We refer to such a scenario as perfect information [30], under which the methodology provided in Section 5.4 will lead to an optimal SLA for the provider. Epistemic uncertainty is manifested through inaccuracy in the estimators generated by a QoS modeling method. The acquisition of perfect information is, for all but the most elementary phenomenon, extremely if not prohibitively costly. Consequently, the provider must accept the ubiquity of epistemic uncertainty, and should seek to minimize this uncertainty to reduce exposure to uncertainty in QoS. These techniques of information gathering to derive a model of the QoS are likely to be utilized in conjunction with control mechanisms such as admission control policies which can be utilized to reduce aleatory uncertainty as much as is feasible given the scope of control of the provider.

Four illustrative QoS modeling methods are defined to demonstrate the difference in epistemic uncertainty experienced through the selection of a QoS modeling method, utilizing different input parameters and deriving varied models of the QoS. The models are assumed to be derive by the QoS modeling methods prior to the formulation of an SLA by the provider. It is important to note that the aim is not to define an optimal QoS modeling method, rather to illustrate the value of QoS modeling in terms of benefit to the provider, where the different methods provide different models and yield different SLAs with different objectives for the provider. The set of feasible QoS modeling methods is vast and may contain methods utilized complex machine learning techniques such that the model of the QoS can be refined over time in accordance with the learning. For the purposes of our work, we omit consideration of such methods, and focus on QoS modeling methods which utilize empirical observations and system state information in a stateless manner to derive the model of the QoS. The methods are assumed to utilize a set, Θ, containing chronologically ordered empirical observations of a given metric obtained from previously completed requests to the service:

$$\Theta = \{\theta_1, \ldots, \theta_m\} \tag{5.5}$$

Additionally, the methods are assumed to have access to system state information pertaining to the number of requests which are in the queue of those to be processed by the service:

$$j \in \mathbb{N} \tag{5.6}$$

In the next four subsections, we define different QoS modeling methods which can derive a model of the QoS from these empirical observations.

5.4.2.1 Random Sampling

The random sampling method derives a model of the QoS through the utilization of historical data selected from Θ based on random sampling. A sample set of size n is drawn from the set of observations, Θ with equal probability, and the size of the sample set, n, can be established in accordance with a variety of criteria, for example, the desired confidence interval. The mean, μ, of the observations in the sample is calculated to provide an estimator for the rate of the exponential distribution, λ:

$$\mu = \frac{\sum \theta_i}{n}, \lambda = \frac{1}{\mu} \tag{5.7}$$

5.4.2.2 Sliding Window

The sliding window method derives a model of the QoS through the utilization of historical data selected from Θ such that the most recent w observations are utilized. The size of the window,

w, can be established in accordance with desired reactivity of the method, with a smaller window denoting increased reactivity to recently completed requests. The mean, μ, of the observations in the sample is calculated to provide an estimator for the rate of the exponential distribution, λ:

$$\mu = \frac{\sum \theta_i}{w}, \lambda = \frac{1}{\mu} \tag{5.8}$$

5.4.2.3 Least Squares

The least squares method derives a model of the QoS through the utilization of historical data selected from Θ in accordance with the sliding window method, and the utilization of statistical inference using Linear Regression (see [207]). The method identifies a correlation between the requests and the QoS utilizing two constants, a and b to represent the correlation, such that the next QoS can be predicted. The least squares line defined by the constants illustrates the linear correlation between the requests and the QoS, and to predict the quality of the next request. This correlation will likely take in account some queueing effect, manifested as a positive correlation.

Let x refer to the request, and y refer to the response time for the request, then parameter b is calculated by:

$$b = \frac{s_{xy}}{s_{xx}} \tag{5.9}$$

where:

$$s_{xy} = \sum xy - \frac{\sum x \sum y}{w+1} \text{ and } s_{xx} = \sum x^2 - \frac{(\sum x)^2}{w+1} \tag{5.10}$$

The parameter a is calculated by:

$$a = \frac{\sum y - b \sum x}{w+1} \tag{5.11}$$

The value calculated by the method is utilized to provide an estimator for the rate of the exponential distribution, λ:

$$\mu = a + b(w+1), \lambda = \frac{1}{\mu}, \lambda = \frac{1}{\mu} \tag{5.12}$$

5.4.2.4 Queue Size

The queue size method derives a model of the QoS through the utilization of historical data selected from Θ in accordance with the sliding window method, and the utilization of information pertaining to the number of requests in the single queue of requests to the service, j, and the number of resources in the QoS r (assumed to serve requests in a sequential manner). The mean of the

empirical observations is calculated, and is assumed to represent an estimator for the value of the given metric for each request in the queue. The queue size, j, and number of resources, r, is then considered, yielding an estimator of the mean, μ. This mean will explicitly consider and react to queuing effects within the QoS, in order to increase the accuracy of the model at any point in time.

$$\mu = \frac{\sum \theta_i}{w} \cdot \left(1 + \frac{j}{r}\right), \lambda = \frac{1}{\mu} \tag{5.13}$$

5.5 Experimental Procedure

In this section, we describe the utilization of the methodology presented in Section 5.4 to fulfill the requirements of the study presented in Section 5.2.

5.5.1 Negative Consequences Of Uncertainty In QoS Model

In order to investigate the negative consequences of epistemic uncertainty in the model of QoS for a Web service, we define an experimental procedure based upon simulation in Maple [208], which demonstrates the change in utility with the change in epistemic uncertainty, under two different set of payment functions.

The QoS under perfect information is modeled utilizing an exponentially distributed random variable, V_m, such that $f_m(v) = exp(\lambda = 0.25)$. The optimal SLA will then be derived utilized the methodology in Section 5.4.1, maximizing Equation (5.2). This SLA will represent the optimal SLA given perfect information regarding the computing infrastructure, in the absence of epistemic uncertainty, and contains an optimal SLO for some value, v of m. In order to investigate the effect of epistemic uncertainty, an error can be introduced into the model of the QoS such that the methodology is utilizing an inaccurate model of the QoS. Accordingly, the process of deriving the optimal SLA will carried out with the introduction of an error into λ, such that modeling error is manifested through an inaccurate value for the rate of the exponential distribution, $\overline{f_m}(v) = exp(\lambda = 0.25 + \epsilon)$ where $\epsilon \in [-0.2, 0.2]$ defines the modeling error, maximizing Equation (5.14). For example, an error of -0.02 would yield a model of the QoS such that $\lambda = 0.23$. This yields an SLA which contains a sub-optimal SLO for some value, $\overline{v}$ of m.

$$EU(m, v) = \int_{x \in V^{+}_{m,v}} \overline{f_m}(x)dx \cdot g_m(v) + \int_{x \in V^{-}_{m,v}} \overline{f_m}(x)dx \cdot h_m(v) \tag{5.14}$$

The expected utility of the value, $\overline{v}$, quoted in the sub-optimal SLA, is then derived under the model of the QoS with perfect information, $EU(m, \overline{v})$. This generates the real expected utility of $\overline{v}$ which will necessarily be less than v. Therefore, the errors in the model which lead to epistemic uncertainty, will necessarily yield sub-optimal SLAs which are potentially damaging to the provider.

The extent of the damage to the provider is determined by the payment functions defined by the provider. Clearly, the greater the payment to the requester in the event of violation, the greater the damage from sub-optimal SLAs which are prone to a higher number of violations. The experimental process is repeated for two sets of indicative payment functions to demonstrate the varied consequences of uncertainty in QoS under different payment functions:

$$\begin{aligned} g_m(v) &= -1.5v + 15 \\ h_m(v) &= 0.5v - 5 \end{aligned} \tag{5.15}$$

and

$$\begin{aligned} g_m(v) &= -1.5v + 15 \\ h_m(v) &= 0 \end{aligned} \tag{5.16}$$

5.5.2 Costs Of QoS Modeling Methods

In order to investigate the cost of the QoS modeling methods which derive the model of the QoS, we define two types of cost which can be incurred and investigate the effect of such costs on the objectives of an organization for a SLA. These costs were utilized in an experimental procedure based upon simulation in Maple [208], which demonstrates the change in utility with the change in cost, under certain payment functions.

5.5.2.1 Constant Cost

A constant cost can be attributed to the derivation of a model by a QoS modeling method which is assigned to each SLA. This changes the expected utility of an SLA for the provider from Equation (5.2) to:

$$EU(m, v, \omega) = \int_{x \in V^+_{m,v}} \overline{f_m}(x)dx \cdot g_m(x) + \int_{x \in V^-_{m,v}} \overline{f_m}(x)dx \cdot (g_m(x) - h_m(x)) - k(\omega) \tag{5.17}$$

The associated of a constant cost, $k(\omega)$, on a per-SLA basis is a challenging undertaking, and would require in-depth information on the scarce resources of the QoS by the QoS modeling method, in order to reflect the opportunity cost. Under the assumption that such a cost could be reasonably assigned, the provider would then select the QoS modeling method whose accuracy is greatest and constant cost is minimized.

Definition 4 (Constant Cost) *A cost for QoS modeling represented by a per-SLA cost function* $k : \Omega \rightarrow \mathbb{R}$ *where* Ω *is the set of QoS modeling methods.*

In order to investigate the effect of constant cost on the expected utility of an SLA for the provider a range of constant costs can be introduced, $k(\omega) \in [0, 2]$, and a benefit from the QoS modeling method, in terms of accuracy, can be introduced in an analogous manner to the experiment in Section 5.5.1. The expected utility is maximized when both constant cost and modeling error are minimized. The constant cost is worth incurring if the resultant accuracy of the model is such that greater expected utility is yielded over and above the cost. As described in Section the negative consequences of uncertainty in QoS are determined by the payment functions. We define a set of payment functions to utilize in this experimental procedure which is analogous to those defined in Equation (5.15).

5.5.2.2 Overhead Cost

An overhead cost can be attributed to the derivation of a model by a QoS modeling method which is assigned to each SLA. The overhead cost emerges through the intrusive effect of the QoS modeling method on the computing infrastructure, caused by the consumption of scarce resources which could be utilized for provision of the service. This expected utility function of the provider remains analogous to Equation (5.2).

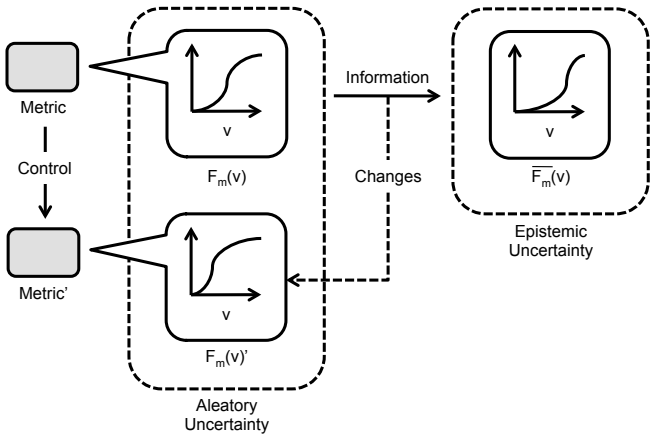

Figure 5.3: An Illustration Of The Overhead Cost Of QoS Modeling Methods

The QoS modeling method, in its utilization of scarce resources, influences the aleatory uncertainty in the QoS, and therefore changes the computing infrastructure it is seeking to model (see Figure 5.3). Therefore, the expected utility of the provider is defined in terms of a distribution, $f'_m(v)$, which results from the intrusive effect of the QoS modeling method. The empirical observations, $\theta_i \in \Theta_m$, utilized by each QoS modeling method will be transformed in accordance with the influence, such that the model utilized to formulate the SLA is directly affected. This has a con-

sequential effect on the expected utility yielded by a given SLA. Thus, not only does the epistemic uncertainty experienced by the QoS modeling method affect expected utility, but also the contribution of the method to aleatory uncertainty made by the method. Accordingly, a QoS modeling method is subject to a trade-off between an effect on epistemic uncertainty and an effect on aleatory uncertainty, and an apparent paradox in which attempts to reduce epistemic uncertainty can increase aleatory uncertainty. The resolution of such a trade-off is beyond the scope of this paper but is an important consideration in the selection of an appropriate QoS modeling method, especially in computing infrastructures with limited resources. A particular challenge is that of decoupling the epistemic uncertainty of the model derived by the QoS modeling method and the intrusive effect of the QoS modeling method on the computing infrastructure, and therefore on aleatory uncertainty.

Definition 5 (Overhead Cost) *A cost for QoS modeling represented by a transformation function* $\delta : f_m(v) \rightarrow f'_m(v)$*, for any all* v*.*

In order to investigate the effect of constant cost on the expected utility of an SLA for the provider a range of overhead costs can be introduced in the interval, $[0, 2]$, and a benefit from the QoS modeling method, in terms of accuracy, can be introduced in an analogous manner to the experiment in Section 5.5.1. The overhead cost in λ represents the intrusive effect of the QoS modeling method, and therefore the aleatory uncertainty resulting from the consumption of scarce resources by the QoS modeling method. Therefore, in the derivation of a model for the computing infrastructure, reflected by a certain value for λ, the provider is changing the underlying value of λ. The process of gathering, processing and filtering information is changing the underlying phenomenon being modeled (see Figure 5.3). For instance, an overhead cost of 0.1 would represent the fact that in the derivation of the value of λ by the QoS modeling method, the actual value of λ is changed in the objective distribution by 0.1, such that the modeling is made on a past state of the QoS. The modeling error in λ represents the epistemic uncertainty resulting from QoS modeling method. We define a set of payment functions to utilize in this experimental procedure which is analogous to those defined in Equation (5.15).

5.5.3 Benefit Of QoS Modeling Methods

In the next two sections, we describe the experimental procedure utilized to evaluate the different QoS modeling methods presented in Section 5.4.2 under different service usage patterns. These methods and service usage patterns were utilized in an experimental procedure based on deployment described in Chapter 4 to demonstrate the change in utility with the change in QoS modeling method, under certain payment functions (analogous to the experimental procedure in Section 5.5.2.2.

In order to investigate the benefit of QoS modeling methods, we define two different service usage patterns: stable usage pattern and unstable usage pattern. The parameters for these service

usage patterns were established utilizing some basic queuing theory. In accordance with the Grid deployment as described in Chapter 4, 20 nodes with a dummy Stock Quote Service installed were utilized, in which the service time (in seconds) for each request was modeled by an exponentially distributed random variable[2], $exp(\lambda)$ where $\lambda = 0.25$. The service time represent the aleatory uncertainty of the service, and can be viewed as modeling the uncertainty in the time taken to execute internal business logic or database queries. The use of the exponential distribution to model the aleatory uncertainty is indicative and not restrictive. The distribution could easily be changed to an alternative distribution such as a normal distribution. Given the service time and the availability of 20 nodes, 5 requests per second can be processed on average. Therefore, the service will be stable when the number of arrivals in a given second is less than 5 and unstable when the number of arrivals in a given second is greater than or equal to 5. We shall refer to 5 jobs per second as 100% service usage. For each different QoS modeling method, and under the two different service usage patterns, the web service was supplied 10000 times by the provider to 100 homogenous requesters who independently requested the service. The random sampling method (see Section 5.4.2.1) was configured to utilize a sample size of $n = 20$ from the set of empirical observations, whilst the sliding window method (see Section 5.4.2.2) was configured to utilize a window size of $w = 20$. This window size was also retained for utilization in the least squares method (see Section 5.4.2.3) and the queue size method (see Section 5.4.2.4).

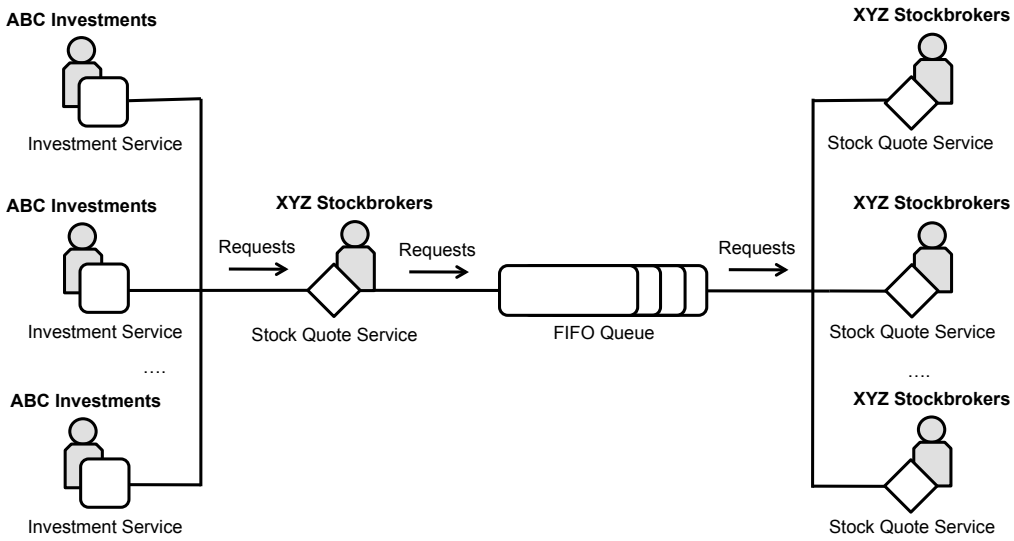

Figure 5.4: An Illustration Of Service Usage Patterns

The requesters were realized as independent threads in an Apache JMeter [209] test plan defined for the experiment. These 100 requesters were introduced gradually over 20 seconds, in order to ramp-up to the specified service usage pattern. The service usage patterns for stable service usage

[2]For each QoS modeling method a bootstrap model where $\lambda = 0.0005$ was utilized. This represents a conservation initial modeling to utilized until subsequent observations are available to refine the modeling.

and unstable service usage are defined as follows:

5.5.3.1 Stable Service Usage

The stable service usage pattern was defined to be indicative of a service usage within the capacity of the service. Accordingly, we established parameters such that the average number of requests arriving in a given time period was equal to the average number of requests being serviced in a given time period. The time between arrivals for the 100 independent requesters was modeled by an exponentially distributed random variable, $exp(\lambda)$ where $\lambda = 0.025$, generating an average of 2.5 requests per second when all 100 requesters were introduced to the provider, which we shall refer to as 50% service usage.

5.5.3.2 Unstable Service Usage

The unstable service usage pattern was defined to be indicative of service usage outside of the capacity of the system such that request were subject to non-negligible time in a queue. We chose to model the time between arrivals for each independent thread by an exponentially distributed random variable, $exp(\lambda)$ where $\lambda = 0.1$, generating an average of 10 requests per second when all 100 requesters were introduced to the provider, which we shall refer to 10 requests per second as 200% service usage. Clearly, such conditions of heavy service usage are not sustainable in the long run for reasons of stability, yet such conditions serve to be indicative of time periods in which we have unanticipated demand which require the system to react.

5.6 Empirical Results And Discussion

In this section, we present and discuss the empirical results obtained from the experimental procedures described in Section 5.5.

5.6.1 Negative Consequences Of Uncertainty In QoS Model

Figure 5.5 illustrates the negative consequences of epistemic uncertainty in the model of the QoS on the expected utility of the provider. As the error in the model generated by the QoS modeling method increases (either positively or negatively), the information utilized to formulate the SLA is increasingly imperfect, and the expected utility decreases as a result of the sub-optimality for the provider of the SLO included in the SLA. The impact on expected utility is particularly significant when it is considered that such sub-optimality would be incurred for each SLA.

The payment functions described in Section 5.4.1 are fundamental to the negative consequences of uncertainty in QoS. The form of these functions defines the monetary payments to which the provider is subject for a violation of the SLO. Greater monetary payments for violation of the SLO

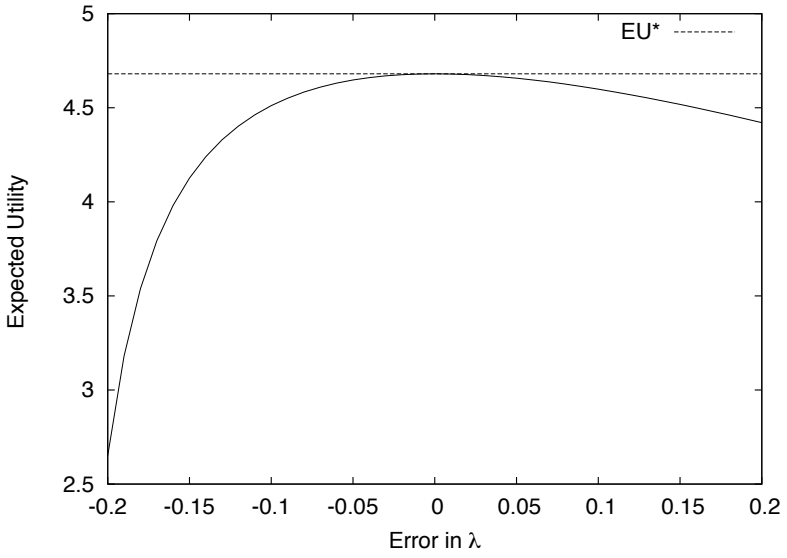

Figure 5.5: A Graph Of Modeling Error vs Expected Utility (a)
$g_m(v) = -1.5v + 15, h_m(v) = 0.5v - 5,\ f_m(v) = exp(\lambda = 0.25)$

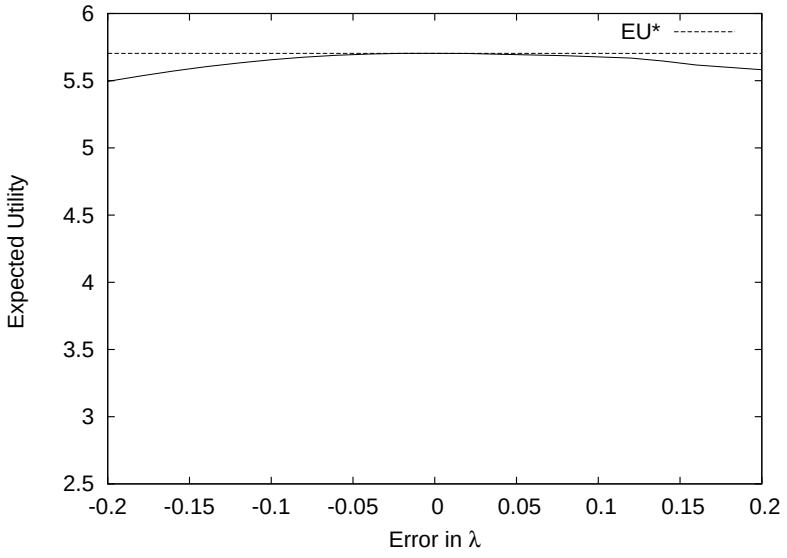

Figure 5.6: A Graph Of Modeling Error vs Expected Utility (b)
$g_m(v) = -1.5v + 15, h_m(v) = 0,\ f_m(v) = exp(\lambda = 0.25)$

will lead the provider to include SLOs in contingent SLAs with a higher probability of fulfillment with a lesser chance of incurring the monetary payments. Accordingly, as shown in Figure 5.2, the provider will not simply quote the mean QoS, due to the fact that the violation of the SLO carries sufficiently high sanctions that the utility of the provider will be negative on violation. The mean QoS is deemed by the provider, under the assumption that the provider is risk-neutral, to yield less utility, in expectation, than an SLO with higher probability.

In the case of different payment functions, the monetary payments to the provider for violation may be reduced, such that the provider expects greater utility from SLOs with lower probability. This is illustrated in Figure 5.6 in which the payment functions are changed such that the monetary payments for violation are reduced to a simple money back guarantee. In doing so the loss in expected utility from errors in modeling are no longer as damaging since uncertainty for the provider, in general, is less damaging. The key point with regard to the payment functions is that the provider is seeking to mitigate the uncertainty in QoS experienced by the requester by taking on that uncertainty. The payment functions are equivalent to a money-back guarantee plus a bonus to the requester for violation of the SLO. Accordingly, regardless of the uncertainty in QoS of the requester, he has an expectation of positive utility from the supply of the service by the provider. Through manipulation of the payment functions, the allocation of uncertainty in QoS to the requester and provider are manipulated which consequently affects the expectation of utility for both. In the case of the payment functions chosen, the negative consequences of the provider are greater due to the uncertainty in QoS he takes on for the requester to increase trust.

5.6.2 Costs Of QoS Modeling Methods

In the next two subsections, we define two different classes of cost which can be associated with the utilization of QoS modeling methods.

5.6.2.1 Constant Cost

Figure 5.7 illustrates the effect of constant cost on the expected utility yielded when utilizing some QoS modeling method. The benefit of a modeling method is manifested in the minimization of modeling error, such that an accurate model of the QoS is derived, and the expectation of utility for a given SLO is not significantly over or under-estimated. Since the cost of a QoS modeling method is manifested in the constant cost of modeling for each SLA, and given an ability to accurately quantify the constant cost of modeling on a per-SLA basis, a simple cost-benefit analysis could be utilized for each method to select that method which resolves the cost-benefit trade-off in the most profitable manner and maximizes the expected utility of the provider.

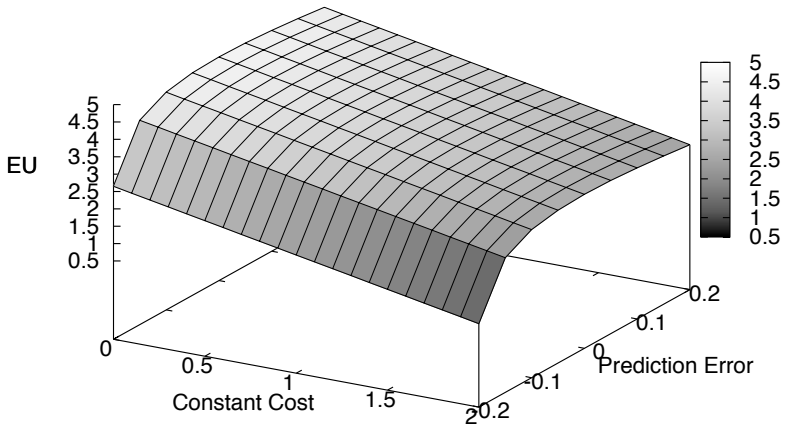

Figure 5.7: A Graph Of Constant Cost vs Modeling Error vs Expected Utility
$g_m(v) = -1.5v + 15, h_m(v) = 0.5v - 5,\ f_m(v) = exp(\lambda = 0.25)$

5.6.2.2 Overhead Cost

Figure 5.8 shows the effect of an overhead cost on the expected utility as derived by the simulation. Perhaps, not intuitively, the increased proximity of the modeling error to the overhead cost reflects increased accuracy in the modeling method. This is due to the fact that the modeling error introduces a certain inaccuracy to the predicted value of λ such that the subjective distribution is inaccurate according to the underlying phenomenon at that point in time. The overhead cost also introduces an inaccuracy to the predicted value of λ such that objective distribution is changed. Therefore, in the case, for example, of a modeling error of 0.1 in Figure 5.8 and an overhead cost of 0.1, the modeling method actually becomes completely accurate with regard to the underlying phenomenon, somewhat inadvertently. Therefore, the closer the modeling error is to the overhead cost, the more utility the QoS modeling method will yield. Expected utility is maximized where modeling error and overhead cost are identical and themselves maximized. Given our payment functions, greater expected utility is yielded by QoS with higher probability and subject to some under-consumption, than QoS with low probability and subject to increased violations and the resultant monetary payments. In a similar manner to constant cost, in the event that the provider is able to decouple the modeling error and overhead cost introduced by a given QoS modeling method, where minimized modeling error is the benefit and overhead cost is the cost, a cost-benefit analysis could again be performed to compare methods and select the method which maximizes expected utility. The provider would

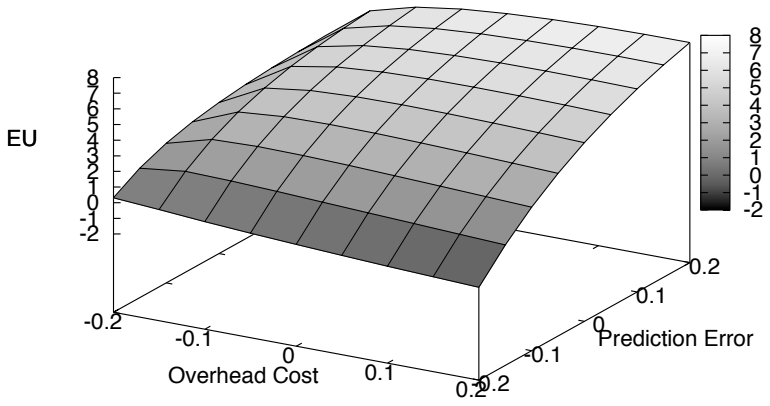

Figure 5.8: An Illustration Of Overhead Cost Of QoS Modeling Methods
$g_m(v) = -1.5v + 15, h_m(v) = 0.5v - 5,\ f_m(v) = exp(\lambda = 0.25)$

desire those QoS modeling method which are minimally intrusive (low overhead cost) and maximally accurate (low modeling error).

5.6.3 Benefit Of QoS Modeling Methods

Figure 5.9 and Figure 5.10 illustrate the utility yielded of the service, utilizing the different QoS modeling methods, and under the different service usage patterns. Under the first service usage pattern (Figure 5.9) all QoS modeling methods yielded similar utility. These results indicate that under steady service usage the QoS modeling method utilized has no significant effect on utility. Under the second service usage pattern, the queue size modeling method yields significantly greater utility than the three other modeling methods, demonstrating the importance of reactivity in the presence of high service usage patterns. The reactivity of the queue size method to the increasing size of the queue under high demand enables the optimal QoS to be adapted accordingly without a significant negative impact on utility. The other methods, such as random sampling and sliding window, require far longer to react to such service usage patterns, relying on information collected after each request, rather than current state information. The utility of the service under the second service usage pattern is positively correlated with reactivity, with least squares method yielding the next greatest utility, followed by sliding window and finally random sampling. Figures 5.11 and 5.12 illustrate the number of the QoS encountered by the service, utilizing the different QoS modeling methods, and under different service usage patterns. In a similar manner to Figures 5.9 and 5.10,

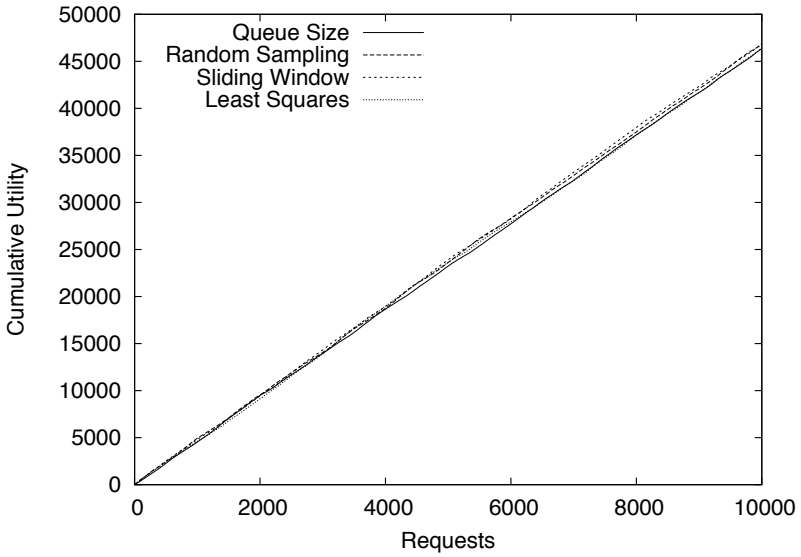

Figure 5.9: A Graph Of Cumulative Utility vs Requests - Stable Service Usage

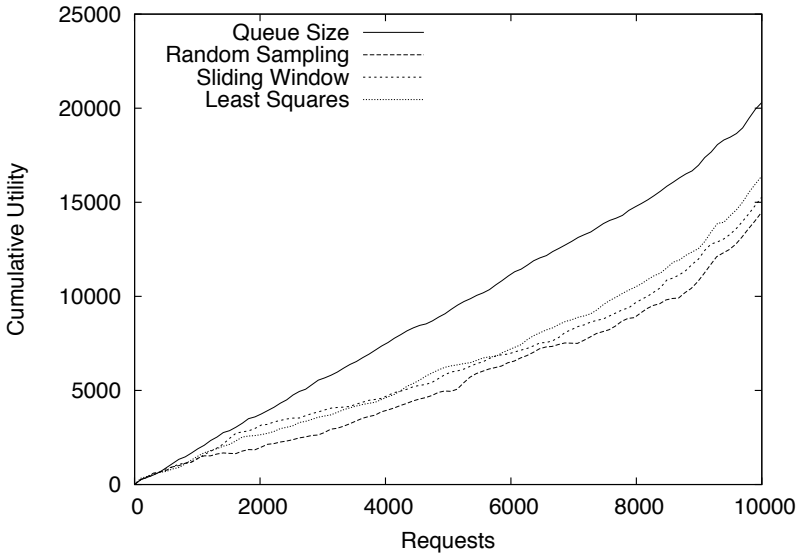

Figure 5.10: A Graph Of Cumulative Utility vs Requests - Unstable Service Usage

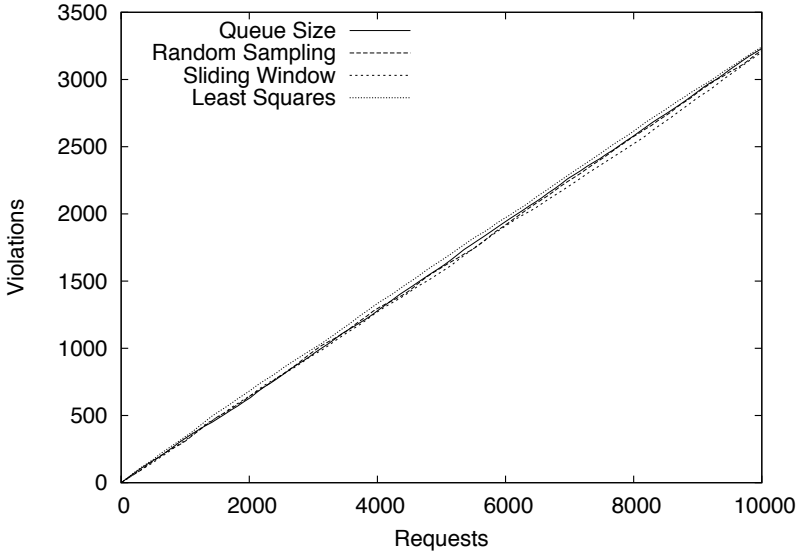

Figure 5.11: A Graph Of Number Of Violations vs Requests - Stable Service Usage

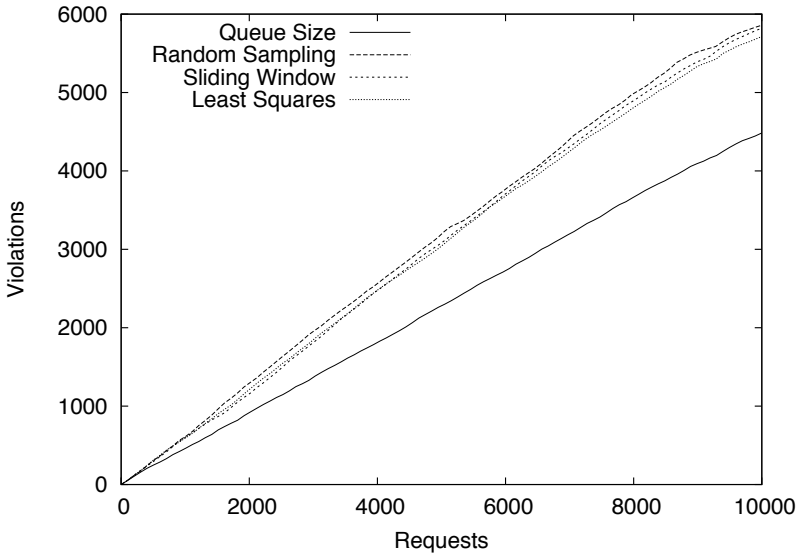

Figure 5.12: A Graph Of Number Of Violations vs Requests - Unstable Service Usage

the reactivity of the queue size method is able to minimize the number of violations under the second service usage pattern. The number of violations impacts directly on utility, in both the present and in the future, through a negative impact on aspects such as trust.

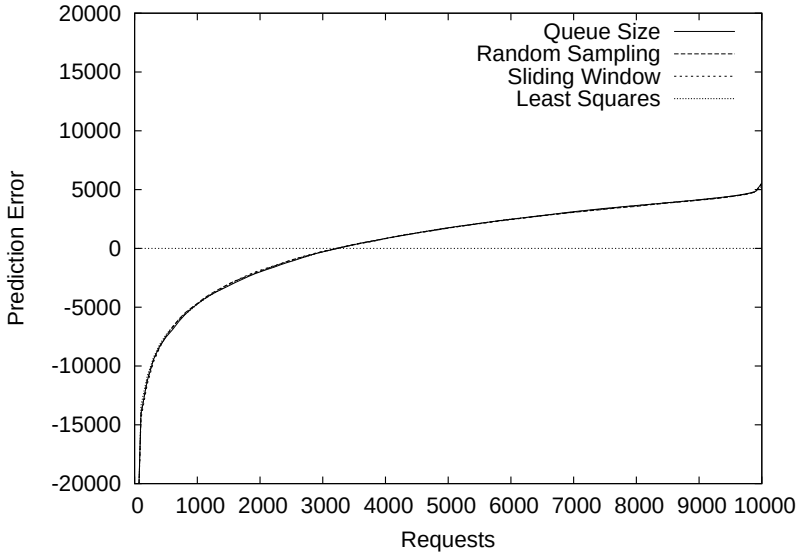

Figure 5.13: A Graph Of Modeling Error vs Requests - Stable Service Usage

	Mean	Std Dev
Queue Size	521.45	4046.51
Random Sampling	517.22	4046.26
Sliding Window	521.07	4058.95
Least Squares	538.26	3988.79

Figure 5.14: A Table Of Modeling Error Statistics - Stable Service Usage

Figure 5.13 and Figure 5.15 illustrate the modeling error experienced by each QoS modeling method. The errors associated with each request are sorted to show the proportion of requests for which there is over-estimation or the quality, and those for which there is under-estimation. Some summary statistics pertaining to these graph are provided in Figure 5.14 and Figure 5.16. It is clear from the graphs that for the 50% service usage there is no significant difference in the error experienced by any of the methods. The difference arises for the 200% service usage, where the computing infrastructure is subject to significant over-utilization. The queue size method is able to adapt to the over-utilization such that the quality predicted reflects the changing state of the infrastructure. These results show that the selection of QoS modeling method can affect the epistemic uncertainty experienced by the provider, although the differentiation of the methods only

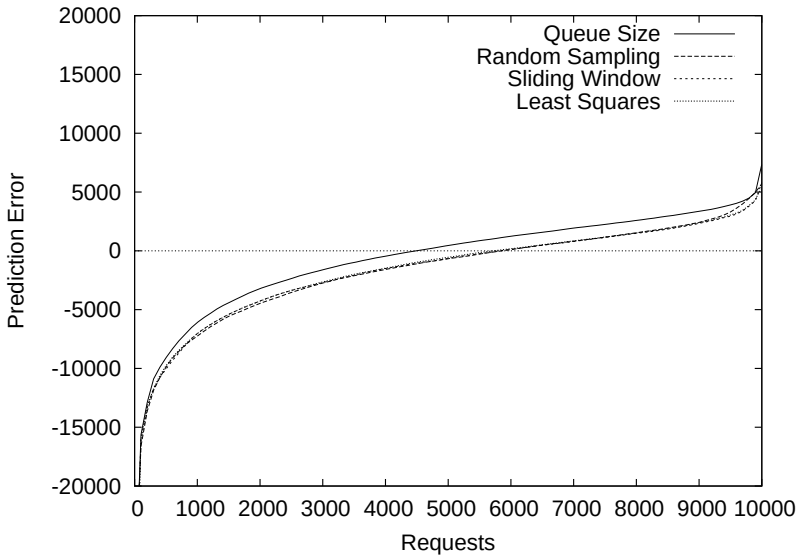

Figure 5.15: A Graph Of Modeling Error vs Requests - Unstable Service Usage

	Mean	Std Dev
Queue Size	-649.69	4285.93
Random Sampling	-1686.38	4255.07
Sliding Window	-1670.58	4159.57
Least Squares	-1623.54	4184.09

Figure 5.16: A Table Of Modeling Error Statistics - Unstable Service Usage

becomes significant under unstable service usage. Those methods utilizing the historical data alone perform similarly under both service usage pattern, with the only differentiation occurring for the queue size modeling method under the unstable service usage pattern. This is due to the fact that under unstable service usage, the queue size method is able to react to the increasing size of the queue and adapt the SLO accordingly.

5.7 Conclusion

This chapter has addressed the problem of uncertainty in QoS experienced by the provider of a Web service. The chapter presents an empirical study of uncertainty in SPRs carried out utilizing the software architecture presented in Chapter 4. The main contribution of the chapter is a theoretical description of a methodology for the creation of SLAs as contingent contracts, whose contingencies are optimal with regard to any organizational objectives, in the presence of uncertainty in the QoS. In the next chapter, we briefly summarize the contributions of this thesis and describe some threads of potential future work.

Chapter 6

Conclusion

The thesis has focused on uncertainty in QoS within a SPR experienced by the requester and provider of the Web service. The thesis has made a number of contributions to address the problem of uncertainty in SPRs, and these contributions have highlighted future threads of work which can contribute further to addressing the problem. This chapter will summarize the contributions of the thesis to the problem of uncertainty in SPRs, and provide a discussion of future threads of work relating to the problem of uncertainty in SPRs which have been highlighted by these contributions.

The remainder of this chapter is structured as follows. Section 6.1 provides a summary of the novel contributions of this thesis. Finally, future work relating to this thesis is discussed in Section 6.2.

6.1 Summary Of Contributions

This thesis has provided a number of key contributions to address the problem of uncertainty in SPRs:

- **A cross-disciplinary background and literature review of uncertainty in SPRs**
  This cross-disciplinary background and literature review summarizes the key concepts and methodologies from a variety of different disciplines including computer science, economics and sociology. Such an approach differs from previous work, which focuses on concepts and methodologies from the perspective of a single discipline. The value of this cross-disciplinary approach is to provide of a theoretical and practical basis for the study of uncertainty in SPRs which jointly considers technological, economic and social perspectives. This enables such perspectives to inform the design and evaluation of software components to support SPRs.

- **An institutional framework for trust in SPRs**
  This framework provides a holistic representation of the environment of an SPR, and the perception of this environment by an organization to derive a notion of trust. Such an approach

differs from previous work, which makes implicit or explicit assumptions regarding the environment without explanation of the implications of such assumptions for trust, and often focuses on environments containing a single class of trust mechanism rather than environments as a holistic encapsulation of the context of an SPR. Additionally, in contrast to previous work, institutions are utilized to model the constraints in the environment of an SPR, such that institutions are perceived by an organization to analyze, evaluate, and decide upon participation in an SPR. The value of this approach is to provide a basis for the evaluation of an environment of an SPRs in terms of the absence or presence of trustworthiness and trust, and the mismatch between trust and trustworthiness. This enables environments to be appropriately designed or manipulated by organizations to increase both trustworthiness and trust.

- **A structured language for the representation of SLAs**
  This structured language enables the representation of SLAs as contingent contracts, whose contingencies are defined over the QoS experienced by the requester. Such an approach differs from previous work, which does not provide a formal basis for the representation of SLAs as contingent contracts, where contingencies are defined in a verifiable and enforceable manner utilizing HTTP status codes. The value of this approach is to enable the utilization of insights from economic literature on the design of contingent contracts to the creation of SLAs, such that the uncertainty of an organization in the QoS can be reduced.

- **A theoretical and practical methodology for the creation of optimal SLAs**
  This methodology facilitates the creation of SLAs as contingent contracts, whose contingencies are optimal with the objectives of an organization, in the presence of uncertainty in QoS. Such an approach differs from previous work, which does not provide a generic process for the creation of optimal SLAs for any QoS metric of a Web service, given organizational objectives, and does not describe the practical methodology required to realize such an optimization process in a software component. The value of such an approach is to provide a formal process for the creation of optimal SLAs, such that organizations no longer utilize processes based on ad-hoc, sub-optimal reasoning which can be inconsistent with organizational objectives.

6.2 Future Work

The work in this thesis can provide the basis for a number of different threads of future work. The different threads of future work which can emerge from each chapter in the thesis can be summarized as follows:

- **An Institutional Framework For Trust In Service Provisioning Relationships**
  The work presented in Chapter 3 highlights a potential future thread of work involving the

empirical evaluation of existing environments for SPRs. This empirical evaluation could consider the trust of a requester in SPRs utilizing the environments, and the trustworthiness of the environment itself. Such an evaluation could demonstrate the mismatches between trust and trustworthiness, highlighting those environments which are trustworthy but not trusted, or vice versa. Additionally, such an evaluation could consider the costs associated with the increasing the trust of a requester in a SPR, the trustworthiness of the environment, or both. The incurrence of such costs by the requester and/or provider in the SPR may not provide a sufficient contribution to trustworthiness and trust to be justified, such as the utilization of a third party Web service for monitoring. Finally, a future thread of work could involve the automated construction of environments for SPRs which are trusted by a requester, under certain assumptions of the disposition, beliefs and objectives of the requester. These beliefs and objectives could be utilized to construct an environment for a SPR which is trusted by the requester in an automated manner, utilizing the set of Web services and communication networks over which the requester has beliefs. Such automated construction may itself be encapsulated within a Web service.

- **A Software Architecture For Service Level Agreements**

  The work presented in Chapter 4 highlights a potential future thread of work involving the expansion of the Web services within the software architecture to include a reputation service. A reputation service of particular interest is that defined in [110]. This reputation service attempts to address the manipulation of reputation data based upon the rationality of organizations in the SPR. For example, the requester may provide false feedback on the QoS in order to reduce the payment required for the utilization of the Web service within the SPR. Given our assumption of rationality for both the requester and provider, such a reputation service would provide control over the provider which both credible for the requester, and viable for the provider. Another potential future thread of work involves the explicit consideration of the impact of the communication network on the QoS of a Web service, and on the ability to control the provider of a Web service with regard to this QoS. The monitoring service within the architecture would need to facilitate the distinction between the impact of the provider of the Web service, and the impact of the communication network on the QoS. In the absence of such consideration, the utilization of Service Level Agreements may incur high costs but not increase the trust sufficiently to make participation in the SPR viable for the requester. Finally, a future thread of work could involve an investigation into the viability of caching of SLAs over a certain period of time. This would avoid the computation involved with optimization on a per-SLA basis, but may introduce additional uncertainty in the QoS, with the computing infrastructure subject to change over the period of caching. In a similar manner, such a thread of work could empirically investigate the computation involved with the monitoring of every

request to a Web service with regard to every metric. For Web services with high load such monitoring may require significant computation which could introduce the uncertainties which are discussed in the empirical study.

- **An Empirical Study Of Uncertainty In Service Provisioning Relationships**
  The work presented in Chapter 5 highlights a potential future thread of work involving the expansion of an SLA beyond a single SLO. In practical SPRs, the requester is likely to be concerned with more than one metric, such as availability and response time. The consideration of multiple metrics would increase the complexity of optimization problem significantly, and may prove to be too costly to perform on a per-SLA basis. Another potential thread of future work involves the utilization of the methodology in conjunction with different admission control mechanisms. Such admission control mechanisms can control the service usage patterns to the extent that the task of QoS modeling is eased. The increase in predictability of the service usage patterns would facilitate the utilization of the methodology to create optimal Service Level Agreements within some usage bounds, providing some security against large fluctuations in service usage which can make the creation of optimal Service Level Agreements subject to high uncertainty. Finally, a future thread of work could involve an investigation into the costs of monitoring and prediction based upon empirical evidence. This would require the ability to apply costs to the consumption of resources by the monitoring and prediction mechanisms, whether in terms of fixed costs from the additional dedicated resources required in the infrastructure, or variable costs in terms of the consumption of existing resources and the consequential effect on QoS.

Bibliography

[1] W3C. Web Services Architecture (WS-ARCH). http://www.w3.org/TR/ws-arch/, February 2004.

[2] Li-Jie Jin, Vijay Machiraju, and Akhil Sahai. Analysis On Service Level Agreements For Web Services. Technical Report HPL-2002-180, HP Laboratories, Palo Alto, USA, June 2002.

[3] Kenneth J. Arrow. The economics of agency. In John W. Pratt and Richard J. Zeckhauser, editors, *Principals and Agents: The Structure of Business*, pages 37–51. Harvard Business School Press, 1985.

[4] J.E. Stiglitz. Principal and agent. Technical Report 12, Princeton, Woodrow Wilson School - Discussion Paper, 1988.

[5] Adam Smith. *An Enquiry into the Nature and Causes of the Wealth of Nations: Book 1*. Methuen and Co. Ltd., 5th edition, 1904.

[6] Yannis Bakos. The emerging role of electronic marketplaces on the internet. *Communications of the ACM*, 41:35–42, 1998.

[7] Feng Li. *What is E-Business? How The Internet Transforms Organizations*. Blackwell, 2006.

[8] Mark Turner, David Budgen, and Pearl Brereton. Turning software into a service. *Computer*, 36(10):38–44, 2003.

[9] Amazon.com Inc. Amazon Elastic Compute Cloud (EC2). http://aws.amazon.com/ec2, 2009.

[10] Microsoft. Microsoft Azure. http://www.microsoft.com/azure/, 2009.

[11] Google Inc. Google Application Engine. http://code.google.com/appengine/.

[12] Michael E. Bratman. *Intentions, Plans and Practical Reason*. Centre for the study of language and information, Stanford, California, 1999.

[13] Michael Wooldridge. *Reasoning about rational agents*. The MIT Press, 2000.

[14] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

[15] Daniel Kahneman. Maps of bounded rationality: Psychology for behavioral economics. *The American Economic Review*, 93(5):1449–1475, 2003.

[16] H. Simon. Theories of bounded rationality. In C. McGuire and R. Radner, editors, *Decision and Organization*, pages 161–176. North Holland, 1972.

[17] W3C. Architecture Of The World Wide Web: Volume One. http://www.w3.org/TR/2004/REC-webarch-20041215/, 2004.

[18] W3C. Uniform Resource Identifier. http://www.w3.org/Addressing/, 2001.

[19] W3C. Hypertext Transfer Protocol (HTTP). http://www.w3.org/Protocols/, 2009.

[20] Morris Sloman, editor. *Network and Distributed Systems Management*. Addison Wesley Publishers Ltd, Wokingham, UK, 1994.

[21] International Organization For Standardization (ISO). Quality Management Systems - Fundamentals and Vocabulary. ISO9000/2005, 2005.

[22] Jeff Tian. Quality-evaluation models and measurements. *IEEE Software*, 21(3):84–91, 2004.

[23] Michael P. Papazoglou. *Web Services: Principles And Technology*. Pearson Education Limited, Harlow, UK, 2008.

[24] Anbazhagan Mani and Arun Nagarajan. Understanding Quality Of Services For Web Services. http://www.ibm.com/developerworks/library/ws-quality.html, 2009.

[25] Daniel A. Menascé. QoS Issues in Web Services. *IEEE Internet Computing*, 6(6):72–75, 2002.

[26] Daniel A. Menascé. Composing Web Services: A QoS View. *IEEE Internet Computing*, 8(6):88–90, 2004.

[27] Aad van Moorsel. Metrics for the Internet Age: Quality of Experience and Quality of Business. Technical Report HPL-2001-179, Software Technology Laboratory, HP Laboratories, Palo Alto, USA, July 2001.

[28] W3C. Web Services Policy (WS-POLICY). http://www.w3.org/TR/ws-policy/, September 2007.

[29] Frank Hyneman Knight and David E. Jones. *Risk, Uncertainty and Profit*. Beard Books, 1st edition, 2002.

[30] Bruce F. Baird. *Managerial Decisions Under Uncertainty.* Wiley Series in Engineering and Technology Management. John Wiley & Sons Inc, New York, USA, 1989.

[31] Amos Tversky and Daniel Kahneman. Judgement under uncertainty: Heuristics and biases. *Science*, 185(4157):1124–1131, September 1974.

[32] J. C. Helton, J. D. Johnson, and W. L. Oberkampf. An exploration of alternative approaches to the representation of uncertainty in model predictions. *Reliability Engineering & System Safety*, 85(1-3):39 – 71, 2004.

[33] Edhuard Hofer. When to separate uncertainties and when not to separate. *Reliability Engineering and System Safety*, 54(1):113–118, 1996.

[34] Sheldon M. Ross. *Introduction to Probability Models.* Academic Press, Inc., 6th edition, 1997.

[35] Ronald Coase. The nature of the firm. *Economica*, 4(16):386–405, 1937.

[36] Oliver E. Williamson. Transaction-cost economics: The governance of contractual relations. *The Journal Of Law & Economics*, 22(2):233–261, 1979.

[37] Francis M. Bator. The anatomy of market failure. *The Quarterly Journal of Economics*, 72(3):351–379, 1958.

[38] Carlos Molina-Jimenez, Jim Pruyne, and Aad van Moorsel. The role of agreements in it management software. In Rogério de Lemos, Cristina Gacek, and Alexander Romanovsky, editors, *Architecting Dependable Systems III*, Lecture Notes in Computer Science, pages 36–58. Springer Berlin / Heidelberg, Berlin, Germany, 2005.

[39] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. Web Services Agreement Specification (WS-Agreement). http://www.ogf.org/documents/GFD.107.pdf, June 2005.

[40] Rashid Al-Ali, Abdelhakim Hafid, Omer Rana, and David Walker. An approach for quality of service adaptation in service-oriented grids: Research articles. *Concurr. Comput. : Pract. Exper.*, 16(5):401–412, 2004.

[41] Karl Czajkowski, Ian Foster, Carl Kesselman, Volker Sander, and Steven Tuecke. Snap: A protocol for negotiating service level agreements and coordinating resource management in distributed systems. In *Proceedings of the 8th Workshop on Job Scheduling Strategies for Parallel Processing*, pages 153–183, 2002.

[42] Viktor Yarmolenko and Rizos Sakellariou. Towards increased expressiveness in service level agreements. *Concurrency and Computation: Practice and Experience*, 19(1):1975–1990, 2007.

[43] James Skene, D. Davide Lamanna, and Wolfgang Emmerich. Precise service level agreements. In *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*, pages 179–188, Washington, DC, USA, 2004. IEEE Computer Society.

[44] Heiko Ludwig, Asit Dan, and Robert Kearney. Cremona: an architecture and library for creation and monitoring of ws-agreements. In *Proceedings of the 2nd international conference on Service oriented computing (ICSOC'04)*, pages 65–74, New York, NY, USA, 2004. ACM.

[45] E.M. Maximilien and M.P. Singh. A framework and ontology for dynamic web services selection. *Internet Computing, IEEE*, 8(5):84–93, Sept.-Oct. 2004.

[46] Chen Zhou, Liang-Tien Chia, and Bu-Sung Lee. DAML-QoS Ontology For Web Services. In *Proceedings of the IEEE International Conference on Web Services (2004)*, San Diego, California, USA, 2004. IEEE Computer Society.

[47] Stephen Marsh and Mark R. Dibben. Trust, untrust, distrust and mistrust - an exploration of the dark(er) side. In Peter Herrmann, Valérie Issarny, and Simon Shiu, editors, *Proceedings of the Third International Conference on Trust Management (iTrust'05)*, volume 3477 of *Lecture Notes in Computer Science*, pages 17–33. Springer, 2005.

[48] Niklas Luhmann. *Trust and Power*. Wiley, 1979.

[49] L. Zucker. Production of trust: Institutional sources of economic structure. *Research in Organizational Behavior*, 6:53–111, 1986.

[50] Annette Baier. Trust and antitrust. *Ethics*, 96(2):231–260, 1986.

[51] T. Grandison and M. Sloman. A survey of trust in internet applications. *IEEE Communication Surveys and Tutorials*, 3, 2003.

[52] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, 2004.

[53] Audun Jøsang. Trust-based decision making for electronic transactions. In *Proceedings of the 4th Nordic Workshop on Secure Computer Systems (NORDSEC'99)*, Stockholm University, Sweden, 1999.

[54] Sviatoslav Brainov and Tuomas Sandholm. Contracting with uncertain level of trust. In *Proceedings of the 1st ACM conference on Electronic commerce (EC'99)*, pages 15–21. ACM, New York, USA, 1999.

[55] Diego Gambetta. Can we trust trust? In Diego Gambetta, editor, *Trust: Making and Breaking Cooperative Relations*, pages 213–237. Department of Sociology, University of Oxford, 1988.

[56] P. Dasgupta. Trust as a commodity. In D. Gambetta, editor, *Trust: Making and Breaking Cooperative Relations*. Blackwell, 2000.

[57] Sarvapali D. Ramchurn, Dong Huynh, and Nicholas R. Jennings. Trust in Multi-Agent Systems. *The Knowledge Engineering Review*, 19(1):1–25, 2004.

[58] D. Harrison McKnight, Larry L. Cummings, and Norman L. Chervany. Initial trust formation in new organizational relationships. *The Academy of Management Review*, 23(3):473–490, July 1998.

[59] D.H. McKnight and N.L. Chervany. What trust means in e-commerce customer relationships: An interdisciplinary conceptual typology. *International Journal of Electronic Commerce*, 6(2):35–53, 2002.

[60] V. Cahill, E. Gray, J.-M. Seigneur, C. Jensen, Y. Chen, B. Shand, N. Dimmock, A. Twigg, J. Bacon, C. English, W. Wagealla, S. Terzis, P. Nixon, G. Serugendo, C. Bryce, M. Carbone, K. Krukow, and M. Nielsen. Using trust for secure collaboration in uncertain environments. *IEEE Pervasive Computing*, 2(3):52–61, September 2003.

[61] Lars Rasmusson and Sverker Jansson. Simulated social control for secure internet commerce. In *Proceedings of the 1996 workshop on New security paradigms (NSPW'96)*, pages 18–25, New York, NY, USA, 1996. ACM.

[62] E. Michael Maximilien and Munindar P. Singh. Toward autonomic web services trust and selection. In *Proceedings of the 2nd International Conference on Service Oriented Computing (ICSOC'04)*, pages 212–221, New York, NY, USA, 2004. ACM.

[63] S. Ketchpel and H. Garcia-Molina. Making Trust Explicit in Distributed Commerce Transactions. In *Proceedings of the 16th International Conference on Distributed Computing Systems (ICDCS '96)*, page 270, Washington, DC, USA, 1996. IEEE Computer Society.

[64] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, 2007.

[65] Debra Meyerson, Karl E. Weick, and Roderick M. Kramer. Swift trust and temporary groups. In Roderick M. Kramer and Tom R. Tyler, editors, *Trust in Organizations: Frontiers of Theory and Research*, pages 166–195. Sage Publications Ltd., 1996.

[66] Jens Riegelsberger, M. Angela Sasse, and John D. McCarthy. The mechanics of trust: a framework for research and design. *International Journal of Human-Computer Studies*, 62(3):381–422, 2005.

[67] Paul A. Pavlou, Yao-Hua Tan, and David Gefen. Institutional trust and familiarity in online interorganizational relationships. In *Proceedings of the 11th European Conference on Information Systems (ECIS'03)*, 2003.

[68] Paul A. Pavlou and David Gefen. Building effective online marketplaces with institution-based trust. *Information Systems Research*, 15(1):37–59, March 2004.

[69] Yao-Hua Tan and Walter Thoen. Formal aspects of a generic model of trust for electronic commerce. *Journal of Decision Support Systems*, pages 233–246, 2002.

[70] Susan P. Shapiro. The social control of impersonal trust. *The American Journal of Sociology*, 93(3):623–658, 1987.

[71] Peter Kollock. The emergence of exchange structures: An experimental study of uncertainty, commitment, and trust. *The American Journal of Sociology*, 100(2):313–345, 1994.

[72] Henning Pagnia, Holger Vogt, and Felix C. Gartner. Fair Exchange. *The Computer Journal*, 46(1):55–75, 2003.

[73] John C. Mitchell and Vanessa Teague. Autonomous nodes and distributed mechanisms. In G. Goos, J. Hartmanis, and J. van Leeuwen, editors, *Software Security: Theories and Systems*, volume 2609 of *Lecture Notes in Computer Science*, pages 295–300. Springer Berlin / Heidelberg, Berlin, Germany, 2003.

[74] Soon-Yong Choi, Andrew B. Whinston, and Dale O. Stahl. Intermediation, Contracts and Micropayments In Electronic Commerce. *Electronic Markets*, 8(1):20–22, 1998.

[75] A. Michael Froomkin. *The Essential Role of Trusted Third Parties in Electronic Commerce*, pages 119–176. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

[76] Paul A. Pavlou. A classification scheme for b2b electronic intermediaries. In M. Warkentin, editor, *Business to Business Electronic Commerce: Challenges and Solutions*. Idea Group Publishing, Hershey, PA, 2002.

[77] Paul Resnick, Richard Zeckhauser, and Chris Avery. Roles for electronic brokers. Working Paper Series 179, MIT Center for Coordination Science, 1998.

[78] Mitrabarun Sarkar, Brian Butler, and Charles Steinfield. Cybermediaries in electronic marketspace: Toward theory building relationships. *Journal of Business Research*, 41(3):215–221, March 1998.

[79] John Darlington, Jeremy Cohen, and William Lee. An architecture for a next-generation internet based on web services and utility computing. In *Proceedings of the 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE '06)*, pages 169–174, Washington, DC, USA, 2006. IEEE Computer Society.

[80] Sean W. Smith. *Trusted Computing Platforms: Design and Applications*. Springer, 2005.

[81] Carlos Molina-Jimenez, Nick Cook, and Santosh Shrivastava. *On the Feasibility of Bilaterally Agreed Accounting of Resource Consumption*, pages 270–283. Springer-Verlag, Berlin, Heidelberg, 2009.

[82] Octavian Catrina and Florian Kerschbaum. Fostering the uptake of secure multiparty computation in e-commerce. In *Proceedings of the 2008 Third International Conference on Availability, Reliability and Security (ARES'08)*, pages 693–700, Washington, DC, USA, 2008. IEEE Computer Society.

[83] S. Chakraborty, S.K. Sehgal, and A.K. Pal. Privacy preserving e-negotiation protocols based on secure multi-party computation. *Proceedings of SoutheastCon 2005.*, pages 455–461, April 2005.

[84] Emerson Ribeiro de Mello, Savas Parastatidis, Philipp Reinecke, Chris Smith, Aad van Moorsel, and Jim Webber. Secure and provable service support for human-intensive real-estate processes. In *Proceedings of the IEEE International Conference on Services Computing (SCC'06)*, pages 495–502, Chicago, USA, September 2006. IEEE Computer Society.

[85] Wenliang Du and Mikhail J. Atallah. Secure multi-party computation problems and their applications: A review and open problems. In *In New Security Paradigms Workshop*, pages 11–20, 2001.

[86] Oded Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2001.

[87] Wembo Mao, Andrew Martin, Hai Jin, and Huanguo Zhang. *Innovations for Grid Security from Trusted Computing*, volume 5087 of *Lecture Notes in Computer Science*, pages 132–149. Springer Berlin / Heidelberg, 2009.

[88] Jeffrey Shneidman, David C. Parkes, and Margo Seltzer. Overcoming rational manipulation in distributed mechanism implementations. Technical Report TR-12-03, Harvard University, 2003.

[89] Jeffrey Shneidman and David C. Parkes. Rationality and self-interest in peer to peer networks. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, 2003.

[90] Jeffrey Shneidman, David C. Parkes, and Laurent Massoulié. Faithfulness in Internet Algorithms. In *Proceedings of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems (PINS'04)*, pages 220–227, New York, NY, USA, 2004. ACM.

[91] Verisign UK Limited. Verisign. http://www.verisign.co.uk, 2009.

[92] Niels Ferguson and Bruce Schneier. *Practical Cryptography.* Wiley, 2003.

[93] Bruce Schneier. *Applied Cryptography.* John Wiley & Sons Inc, New York, USA, 1996.

[94] R. Housley, W. Ford, W. Polk, and D. Solo. Internet X.509 Public Key Infrastructure. http://www.ietf.org/rfc/rfc2459.txt, January 1999.

[95] Audun Jøsang and Simon Pope. Semantic constraints for trust transitivity. In *Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling (APCCM'05)*, pages 59–68, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.

[96] Matt Blaze, Joan Feigenbaum, John Ioannidis, and Angelos D. Keromytis. *The role of trust management in distributed systems security*, pages 185–210. Springer-Verlag, London, UK, 1999.

[97] Dan J. Kim, Donald L. Ferrin, and H. Raghav Rao. A trust-based consumer decision-making model in electronic commerce: The role of trust, perceived risk, and their antecedents. *Decision Support Systems*, 44(2):544 – 564, 2008.

[98] Cristiano Castelfranchi and Rino Falcone. Trust is much more than subjective probability: Mental components and sources of trust. In *Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS'00)*, page 6008, Washington, DC, USA, 2000. IEEE Computer Society.

[99] Sulin Ba and Paul Pavlou. Evidence of the effect of trust building technology in electronic markets: Price premiums and buyer behavior. *MIS Quarterly*, 26(3):243–268, September 2002.

[100] Roy J. Lewicki and Barbara Benedict Bunker. Developing and maintaining trust in work relationships. In Roderick M. Kramer and Tom R. Tyler, editors, *Trust in Organizations: Frontiers of Theory and Research*, pages 114–139. Sage Publications Ltd., 1996.

[101] Rino Falcone and Cristiano Castelfranchi. Social trust: a cognitive approach. In *Trust and deception in virtual societies*, pages 55–90. Kluwer Academic Publishers, Norwell, MA, USA, 2001.

[102] Reinhard Bachmann. Trust - conceptual aspects of a complex phenomenon. In Christel Lane and Reinhard Bachmann, editors, *Trust Within And Between Organizations: Conceptual Issues and Empirical Applications*, pages 298–320. Oxford University Press, 2001.

[103] Bernardo A. Huberman and Fang Wu. The dynamics of reputations. *Journal of Statistical Mechanics: Theory and Experiment*, 2004(04):P04006, 2004.

[104] Philip Nelson. Advertising as information. *Journal of Political Economy*, 82(4):729–54, July/Aug. 1974.

[105] Amazon.com Inc. Amazon. http://www.amazon.co.uk, 2009.

[106] eBay Inc. ebay. http://www.ebay.co.uk, 2009.

[107] P Resnick and R Zeckhauser. Trust amongst strangers in internet transactions: Empirical analysis of ebay's reputation mechanism. *The Economics of the Internet and E-Commerce*, 2002.

[108] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[109] Google Inc. Google. http://www.google.co.uk.

[110] Radu Jurca and Boi Faltings. Reputation-based service level agreements for web services. In Boualem Benatallah, Fabio Casati, and Paolo Traverso, editors, *Service-Oriented Computing - ICSOC 2005*, volume 3826 of *Lecture Notes in Computer Science*, pages 396–409. Springer Berlin / Heidelberg, Berlin, Germany, 2005.

[111] Jordi Sabater and Carles Sierra. Regret: reputation in gregarious societies. In *Proceedings of the fifth international conference on Autonomous agents (AGENTS '01)*, pages 194–195, New York, NY, USA, 2001. ACM.

[112] Ziqiang Xu, P. Martin, W. Powley, and F. Zulkernine. Reputation-Enhanced QoS-based Web Services Discovery. In *Proceedings of the IEEE International Conference on Web Services, 2007 (ICWS'07).*, pages 249–256, July 2007.

[113] Ali Shaikh Ali, Simone A. Ludwig, and Omer F. Rana. A cognitive trust-based approach for web service discovery and selection. In *Proceedings of the Third European Conference on Web Services (ECOWS'05)*, page 38, Washington, DC, USA, 2005. IEEE Computer Society.

[114] Inés Macho-Stadler and J. David Pérez-Castrillo. *An Introduction to the Economics of Information: Incentives and Contracts*. Oxford University Press, Oxford, UK, 1997.

[115] James Skene, Allan Skene, Jason Crampton, and Wolfgang Emmerich. The monitorability of service-level agreements for application-service provision. In *Proceedings of the 6th international workshop on Software and performance (WOSP'07)*, pages 3–14, New York, NY, USA, 2007. ACM.

[116] Rajdeep K. Dash, Nicholas R. Jennings, and David C. Parkes. Computational-mechanism design: A call to arms. *IEEE Intelligent Systems*, 18(6):40–47, November/December 2003.

[117] Joan Feigenbaum and Scott Shenker. Distributed algorithmic mechanism design: Recent results and future directions. In *Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 1–13. ACM Press, 2002.

[118] Noam Nisan. Algorithmic mechanism design. *Games and Economic Behaviour*, 35(1):166–196, April 2001.

[119] Tuomas W. Sandholm. Distributed rational decision making. In G. Weiß, editor, *A Modern Introduction to Distributed Artificial Intelligence*, pages 201–258. MIT Press, Massachusetts, USA, 1999.

[120] Hal R. Varian. Economic mechanism design for computerized agents. In *Proceedings of the 1st USENIX workshop on Electronic Commerce*, pages 13–21, July 1995.

[121] Roger B. Myerson. *Game Theory: Analysis Of Conflict*. Harvard University Press, Cambridge, MA, 1991.

[122] Dov Monderer and Moshe Tennenholtz. Distributed games: From mechanisms to protocols. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence conference (AAAI '99/IAAI '99)*, pages 32–37, 1999.

[123] Edward H. Clarke. Multipart pricing of public goods. *Public Choice*, 11(1):17–33, September 1971.

[124] Theodore Groves and John Ledyard. Optimal allocation of public goods: A solution to the 'free rider problem'. Discussion Papers 144, Northwestern University, Center for Mathematical Studies in Economics and Management Science, March 1976.

[125] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

[126] Jr. Nash, John F. The bargaining problem. *Econometrica*, 18(2):155–162, 1950.

[127] Roger B. Myerson. Incentive compatibility and the bargaining problem. *Econometrica*, 47(1):61–74, January 1979.

[128] Roger B. Myerson and Mark A. Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of Economic Theory*, 29(2):265–281, April 1983.

[129] O. Hart and B. Holmström. The theory of contracts. In Truman Bewley, editor, *Advances in Economic Theory, Fifth World Congress*. Cambridge University Press, 1987.

[130] Bernard Salanié. *The Economics of Contracts: A Primer*. MIT Press, Massachusetts, USA, 1997.

[131] Jean-Jacques Laffont and David Martimort. *The Theory of Incentives: The Principal-Agent Model*. Princeton University Press, 2001.

[132] Hemant K. Bhargava and Shankar Sundaresan. Managing quality uncertainty through contingency pricing. In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS '03)*, Washington, DC, USA, 2003. IEEE Computer Society.

[133] Debasis Mitra Eric Bouillet and K.G. Ramakrishnan. The structure and management of service level agreements in networks. *IEEE Journal on Selected Areas in Communication*, 20(4):691–699, 2002.

[134] Bernardo A. Huberman, Fang Wu, and Li Zhang. Ensuring trust in one time exchanges: solving the QoS problem. *Netnomics*, 7(1):27–37, 2005.

[135] Zoran Milosevic and R. Geoff Dromey. On expressing and monitoring behaviour in contracts. In *Proceedings of the 6th International Enterprise Distributed Object Computing Conference (EDOC'02)*, pages 3–14, Washington, DC, USA, 2002. IEEE Computer Society.

[136] Carlos Molina-Jimenez, Santosh Shrivastava, Jon Crowcroft, and Panos Gevros. On the monitoring of contractual service level agreements. In *Proceedings of the First IEEE International Workshop on Electronic Contracting (WEC '04)*, pages 1–8. IEEE Computer Society, 2004.

[137] N Asokan. *Fairness in Electronic Commerce*. PhD thesis, School of Computer Science, University of Waterloo, 1998.

[138] Holger Vogt, Henning Pagnia, and Felix C. Gartner. Modular fair exchange protocols for electronic commerce. In *Proceedings of the 15th Annual Computer Security Applications Conference*, pages 3–11. IEEE Computer Society Press, 1999.

[139] Levente Buttyán and Jean-Pierre Hubaux. Rational exchange - a formal model based on game theory. In *Proceedings of the 2nd International Workshop on Electronic Commerce (WELCOM'01)*, pages 114–126, London, UK, 2001. Springer-Verlag.

[140] Tuomas Sandholm and XiaoFeng Wang. (Im)possibility of safe exchange mechanism design. In *Eighteenth National Conference On Artificial intelligence*, pages 338–344, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[141] Robert McGrew and Yoav Shoham. Using contracts to influence the outcome of a game. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI'04)*, pages 238–244, San Jose, California, USA, 2004. The MIT Press.

[142] The National Conference Of Commissioners On Uniform State Laws. Uniform electronic transactions act. http://www.nccusl.org/Update/uniformactp, 1999.

[143] Efraim Turban, Dave King, Jae Kyu Lee, and Dennis Viehland. *Electronic Commerce: A Managerial Perspective*. Prentice Hall, 2006.

[144] Hagit Attiya and Jennifer Welsh. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. John Wiley & Sons Inc, Hoboken, USA, 2nd edition edition, 2004.

[145] A. Westerinen et al. Terminology for Policy-based Management. http://www.ietf.org/rfc/rfc3198.txt, November 2001.

[146] A. Dan, D. Davis, R. Kearney, A. Keller, R. King, D. Kuebler, H. Ludwig, M. Polan, M.Spreitzer, and A. Youssef. Web services on demand: Wsla-driven automated management. *IBM Systems Journal*, 43(1):136–158, 2004.

[147] Jurgen Schönwälder, Aiko Pras, and Jean-Philippe Martin-Flatin. On the Future of Internet Management Technologies. *IEEE Communications Magazine*, 2003.

[148] Jeroen van Sloten, Aiko Pras, and Marten van Sinderen. On the standardisation of Web Service management operations, 2004.

[149] J.D. Case, M. Fedor, M.L. Schoffstall, and J. Davin. Simple Network Management Protocol (SNMP). http://www.ietf.org/rfc/rfc1157.txt, May 1990.

[150] William Vambenepe. Web Services Distributed Management: Management Using Web Services (MUWS 1.0) Part 1. http://docs.oasis-open.org/wsdm/2004/12/cd-wsdm-muws-part1-1.0.pdf, 2005.

[151] DMTF. Web-Based Enterprise Management (WBEM). http://www.dmtf.org/standards/wbem/, October 2005.

[152] L. Cherkasova and P. Phaal. Session-based admission control: a mechanism for peak load management of commercial web sites. *Computers, IEEE Transactions on*, 51(6):669–685, Jun 2002.

[153] Heiko Ludwig. Web Services QoS: External SLAs and Internal Policies Or: How Do We Deliver What We Promise? In *Fourth International Conference on Web Information Systems Engineering Workshops (WISEW '03)*, pages 115–120, 2003.

[154] Ali Afzal, A. Stephen McGough, and John Darlington. Capacity planning and scheduling in grid computing environments. *Future Generation Computer Systems*, 24(5):404 – 414, 2008.

[155] Jia Yu and Rajkumar Buyya. A taxonomy of workflow management systems for grid computing. *Journal of Grid Computing*, 3:171–200(30), September 2005.

[156] Rajkumar Buyya. *Economic-based Distributed Resource Economic-based Distributed Resource Management and Scheduling for Grid Computing.* PhD thesis, School of Computer Science and Software Engineering, Monash University, Melbourne, Australia, April 2002.

[157] Florentina I. Popovici and John Wilkes. Profitable services in an uncertain world. In *Proceedings of the 2005 ACM/IEEE conference on Supercomputing (SC '05)*, page 36, Washington, DC, USA, 2005. IEEE Computer Society.

[158] Chee Shin Yeo and R. Buyya. Integrated risk analysis for a commercial computing service. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium, 2007 (IPDPS '07)*, pages 1–10, March 2007.

[159] G. A. Paleologo. Price-at-risk: A methodology for pricing utility computing services. *IBM Systems Journal*, 43(1):20–31, 2004.

[160] J. Palmer and I. Mitrani. Optimal and heuristic policies for dynamic server allocation. *Journal of Parallel and Distributed Computing*, 65(10):1204 – 1211, 2005. Design and Performance of Networks for Super-, Cluster-, and Grid-Computing Part I.

[161] Joris Slegers, Isi Mitrani, and Nigel Thomas. Evaluating the optimal server allocation policy for clusters with on/off sources. *Performance Evaluation*, 66(8):453 – 467, 2009. Selected papers of the Fourth European Performance Engineering Workshop (EPEW) 2007 in Berlin.

[162] Douglass C. North. *Institutions, Institutional Change and Economic Performance.* Cambridge University Press, 1990.

[163] Frank Dignum. Abstract norms and electronic institutions. In *Proceedings of the International Workshop on Regulated Agent-Based Social Systems: Theories and Applications (RASTA'02)*, pages 93–104, 2002.

[164] John R. Commons. *The Legal Foundations of Capitalism.* A. M. Kelley, 1974.

[165] Yoram Moses and Moshe Tennenholtz. Artificial social systems. *Computers and AI*, 14:533–562, 1995.

[166] Robert Axelrod and Daniel O. Keohane. Acheiving Cooperation under Anarchy: Strategies And Institutions. *World Politics*, 38(1):226–254, October 1985.

[167] Harrison C. White. *Identity And Control: A Structural Theory Of Social Action*. Princeton University Press, 1992.

[168] Grahame Thompson, Jennifer Frances, Rosalind Levačić, and Jeremy Mitchell, editors. *Markets, Hierarchies and Networks: The Coordination Of Social Life*. Sage Publications Ltd, 1991.

[169] Robert Axelrod. *The Evolution Of Cooperation*. Basic Books Ltd, 1984.

[170] Thomas C. Schelling. *The Strategy Of Conflict*. Harvard University, Cambridge, Massachusetts, USA, 1997.

[171] Randell L. Calvert. The rational choice theory of social institutions: Cooperation, coordination, and communication. In Jeffrey S. Banks, editor, *Modern Political Economy: Old Topics, New Directions*. Cambridge University Press, 1995.

[172] Richard W. Scott. *Institutions and Organizations (Foundations for Organizational Science)*. Sage Publications Ltd, 2nd edition, 2000.

[173] Marc Esteva, Bruno Rosell, Juan A. Rodríguez-Aguilar, and Josep Ll. Arcos. Ameli: An agent-based middleware for electronic institutions. *Autonomous Agents and Multiagent Systems, International Joint Conference on*, 1:236–243, 2004.

[174] Nicoletta Fornara and Marco Colombetti. Specifying and enforcing norms in artificial institutions. In *AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 1481–1484, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.

[175] Javier Vazquez-Salceda. *The Role of Norms and Electronic Institutions in Multi-Agent Systems Applied To Complex Domains: The Harmonia Framework*. PhD thesis, Computer Science Department, Technical University of Catalonia, April 2003.

[176] Ian Foster and Carl Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[177] J. L. Abad Peiro, N. Asokan, M. Steiner, and M. Waidner. Designing a generic payment service. *IBM Systems Journal*, 37(1):72–88, 1997.

[178] N. Asokan, Philippe A. Janson, Michael Steiner, and Michael Waidner. The state of the art in electronic payment systems. *IEEE Computer*, 30(9):28–35, 1997.

[179] Jeremy Cohen, John Darlington, and William Lee. Payment and negotiation for the next generation grid and web. *Concurrency and Computation: Practice and Experience*, 20(3):239–251, May 2007.

[180] Joan Feigenbaum, Christos Papadimitriou, Rahul Sami, and Scott Shenker. Incentive-compatible interdomain routing. In *Proc. of the 7th Conference on Electronic Commerce (ECí06), 2006*, pages 130–139. ACM Press, 2006.

[181] Drew Fudenburg and Jean Tirole. *Game Theory*. MIT Press, Massachusetts, USA, 1992.

[182] Thráinn Eggertsson. *Economic Behavior and Institutions*. Cambridge University Press, 1990.

[183] Max H. Bazerman and James J. Gillespie. Betting on the future: The virtues of contingent contracts. *Harvard Business Review*, pages 155–160, September-October 1999.

[184] Dewayne E. Perry and Alexander L. Wolf. Foundations for the study of software architecture. *ACM SIGSOFT Software Engineering Notes*, 17(4):40–52, 1992.

[185] Len Bass, Paul Clements, and Rick Kazman. *Software architecture in practice*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998.

[186] Robert B. Grady and Deborah L. Caswell. *Software metrics: establishing a company-wide program*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1987.

[187] J. McCall, P. Richards, and G. Walters. Factors in software quality. Technical Report NTIS AD-A049-014, 015, 055, National Technical Information Service, November 1977.

[188] Fischer Black. The pricing of commodity contracts. *Journal of Financial Economics*, 3(1-2):167 – 179, 1976.

[189] Roger S. Pressman. *Software Engineering: A Practioner's Approach*. McGraw Hill, 5th edition, 2000.

[190] Roy T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000.

[191] Eric Newcomer and Greg Lomow. *Understanding SOA with Web services*. Addison-Wesley, 2005.

[192] Oasis Open. Reference Model For Service Oriented Architecture. http://www.oasis-open.org/committees/soa-rm/, October 2006.

[193] Facebook Inc. Facebook. http://www.facebook.co.uk.

[194] OpenID Foundation. OpenID. http://openid.net/, 2009.

[195] Paypal. Paypal. http://www.paypal.co.uk, 2009.

[196] Western Union International Limited. Western union. http://www.westernunion.co.uk.

[197] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Pearson Education Limited, Harlow, UK, 2nd edition edition, 2001.

[198] J. Klensin. A Universally Unique Identifier (UUID) URN Namespace. http://www.ietf.org/rfc/rfc4122.txt, July 2005.

[199] IETF. Javascript Object Notation (JSON). http://www.ietf.org/rfc/rfc4627.txt, July 2006.

[200] Inc Sun Microsystems. Java. http://java.sun.com, 2009.

[201] Apache Software Foundation. Apache Tomcat. http://tomcat.apache.org/, 2009.

[202] Sun Microsystems. Java Servlet Technology. http://java.sun.com/products/servlet/, 2009.

[203] Apache Software Foundation. Apache Derby. http://db.apache.org/derby/, 2009.

[204] XStream. XStream. http://xstream.codehaus.org, 2009.

[205] Apache Software Foundation. Apache Xerces. http://xerces.apache.org, 2009.

[206] Amazon.com Inc. Amazon Simple Storage Service (S3). http://aws.amazon.com/s3, 2009.

[207] Louise Swift and Sally Piff. *Quantitative Methods for Business, Management and Finance*. Palgrave Macmillan, New York, USA, 2nd edition, 2005.

[208] MapleSoft. Maple. http://www.maplesoft.com, 2009.

[209] Apache Software Foundation. Apache JMeter. http://jakarta.apache.org/jmeter, 2009.